

Numerical Methods for Linear Control Systems



Biswa Nath Datta

```

function [GA,fla]=fun
% [GA,flag]=allo % 
sga=length (vs); sga
if (nargin < 2) tolif (
Ga=zeros (sga,sga)
for k=1:sga-1    for
imsk=imag
if abs(imsk)
if flag==0
if flag==1
flag=mod
end
end
end
end
Ga (sga,sga)=real(Ga)
if flag==0      if f
imsk=imag(vs)
if abs(imsk) >
flag=1;
end
else
else
Ga(sga-1,sga)=
end
end
return % of alloretu

```



**CD-ROM
included**
*Containing
MATLAB-based
solutions
software*

NUMERICAL METHODS FOR LINEAR CONTROL SYSTEMS DESIGN AND ANALYSIS

B.N. Datta

Department of Mathematical Sciences
Northern Illinois University
DeKalb, IL 60115
e-mail: dattab@math.niu.edu

March 10, 2003

Contents

PREFACE	17
1 INTRODUCTION AND OVERVIEW	1
1.1 Linear and Numerical Linear Algebra (Chapter 2 and Chapters 3 and 4)	4
1.2 System Responses (Chapter 5)	7
1.3 Controllability and Observability problems (Chapter 6)	9
1.4 Stability and Inertia (Chapter 7)	10
1.5 Lyapunov, Sylvester, and Algebraic Riccati Equations (Chapter 8 and Chapter 13)	11
1.6 Realization and Identification (Chapter 9)	14
1.7 Feedback Stabilization and Eigenvalue Assignment (Chapter 10 and Chapter 11) .	15
1.8 State Estimation (Chapter 12)	17
1.9 Internal Balancing and Model Reduction (Chapter 14)	19
1.10 Krylov Subspace Methods for Large and Sparse Control Problems (Chapter 15) .	20
1.11 Numerical Methods for Control Problems Modeled by Systems of Second-Order Differential Equations (Chapter 16)	20
1.12 Nearness to Uncontrollability and Instability (Chapter 6 and Chapter 7) and Robust Stability and Stability Radius (Chapter 7 and Chapter 10)	22
1.12.1 Nearness to Uncontrollability and Instability	22
1.12.2 Robust stability and stability Radius (Chapter 7 and Chapter 10)	23
1.13 Sensitivity and Condition Numbers of Control Problems	24
1.13.1 Conditioning of the Matrix Exponential Problem (Chapter 5)	25
1.13.2 Conditioning of the Lyapunov and Sylvester Equations (Chapter 8)	25
1.13.3 Conditioning of the Algebraic Riccati Equations (Chapter 13)	26
1.13.4 Conditioning of the Feedback and Eigenvalue Assignment Problems (Chapter 11)	27
1.14 H_∞ -Control (Chapter 10)	27
1.15 Software for Control Problems	29

2 A REVIEW OF SOME BASIC CONCEPTS AND RESULTS FROM THEORETICAL LINEAR ALGEBRA	33
2.1 Introduction	35
2.2 Vectors	35
2.2.1 Orthogonality of Vectors and Subspaces \mathbb{R}	36
2.3 Matrices	36
2.3.1 Basic Concepts	36
2.3.2 Range and Nullspaces	39
2.3.3 Rank of a Matrix	39
2.3.4 The Inverse of a Matrix	40
2.3.5 The Generalized Inverse of a Matrix	41
2.3.6 Similar Matrices	41
2.3.7 Orthogonal Projection	41
2.4 Some Special Matrices	42
2.4.1 Diagonal and Triangular Matrices	42
2.4.2 Unitary (Orthogonal) Matrix	42
2.4.3 Permutation Matrix	43
2.4.4 Hessenberg (Almost Triangular) Matrix	43
2.4.5 Companion Matrix	44
2.4.6 Nonderogatory Matrix	44
2.4.7 The Jordan Canonical Form of a Matrix	45
2.4.8 Positive Definite Matrix	45
2.4.9 Block Matrices	46
2.5 Vector and Matrix Norms	47
2.5.1 Vector Norms	47
2.5.2 Matrix Norms	48
2.6 Norm Invariant Properties Under Unitary Matrix Multiplication	50
2.7 Kronecker Product, Kronecker Sum and Vec Operation	51
3 SOME FUNDAMENTAL TOOLS AND CONCEPTS FROM NUMERICAL LINEAR ALGEBRA	59
3.1 Introduction	61
3.2 Floating Point Numbers and Errors in Computations	61
3.2.1 Floating Point Numbers	61
3.2.2 Rounding Errors	62
3.2.3 Laws of Floating Point Arithmetic	63
3.2.4 Catastrophic Cancellation	64
3.3 Conditioning, Efficiency, Stability and Accuracy	64
3.3.1 Algorithms and Pseudocodes	64
3.3.2 Solving an Upper Triangular System	65
3.3.3 Solving a Lower Triangular System	65

3.3.4	Efficiency of an Algorithm	66
3.3.5	The Concept of Numerical Stability	66
3.3.6	Conditioning of the Problem and Perturbation Analysis	68
3.3.7	Conditioning of the Problem, Stability of the Algorithm, and Accuracy of the Solution	69
3.3.8	Conditioning of the Linear System and Eigenvalue Problems	69
3.4	LU Factorization	73
3.4.1	LU Factorization using Gaussian Elimination	73
3.4.2	The Cholesky Factorization	78
3.4.3	LU Factorization of an Upper Hessenberg Matrix	79
3.5	Numerical Solution of the Linear System $Ax = b$	81
3.5.1	Solving $Ax = b$ using the inverse of A	81
3.5.2	Solving $Ax = b$ using Gaussian Elimination with Partial Pivoting	81
3.5.3	Solving a Hessenberg Linear System	83
3.5.4	Solving $AX = B$	83
3.5.5	Finding the Inverse of A	83
3.5.6	Computing the Determinant of A	83
3.5.7	Iterative Refinement	84
3.6	The QR Factorization	84
3.6.1	Householder Matrices	85
3.6.2	The Householder QR Factorization	85
3.6.3	Givens Matrices	88
3.6.4	The QR Factorization Using Givens Rotations	90
3.6.5	The QR Factorization of a Hessenberg Matrix Using Givens Matrices . .	91
3.7	Orthonormal Bases and Orthogonal Projections Using QR Factorization	91
3.8	The Least-Squares Problem	92
3.8.1	Solving the Least-Squares Problem Using Normal Equations	92
3.8.2	Solving the Least-Squares Problem Using QR Factorization	93
3.9	The Singular Value Decomposition (SVD)	94
3.9.1	The Singular Value Decomposition and the Structure of a Matrix	96
3.9.2	Orthonormal Bases and Orthogonal Projections	97
3.9.3	The Rank and the Rank-Deficiency of a Matrix	98
3.9.4	Numerical Rank	99
3.9.5	Solving the Least Squares Problem Using the Singular Value Decomposi- tion	100
3.10	Summary and Review	102
3.11	Chapter Notes and Further Reading	105

4 CANONICAL FORMS OBTAINED VIA ORTHOGONAL TRANSFORMATIONS	107
4.1 Importance and Significance of Using Orthogonal Transformations	108
4.2 Hessenberg Reduction of a Matrix	110
4.2.1 Uniqueness in Hessenberg Reduction: The Implicit Q Theorem	111
4.3 The Real Schur Form of A : The QR Iteration Method	112
4.3.1 The Basic QR Iteration	113
4.3.2 The Hessenberg QR Iteration and Shift of Origin	113
4.3.3 The Double Shift QR Iteration	114
4.3.4 Obtaining the Real Schur Form A	115
4.3.5 The Real Schur Form and Invariant Subspaces	117
4.4 Power Iteration and Inverse Iteration	119
4.5 Computing the Singular Value Decomposition	120
4.6 The Generalized Real Schur Form: The QZ algorithm	122
4.6.1 Reduction to Hessenberg-Triangular Form	124
4.6.2 Reduction to the Generalized Real Schur Form	126
4.7 Computing of the Eigenvectors of the Pencil $A - \lambda B$	128
4.8 Summary and Review	129
4.9 Chapter Notes and Further Reading	131
5 LINEAR STATE SPACE MODELS AND SOLUTIONS OF THE STATE EQUATIONS	135
5.1 Introduction	136
5.2 State-Space Representations of Control Systems	137
5.2.1 Continuous-Time Systems	137
5.2.2 Discrete-Time Systems	148
5.2.3 Descriptor Systems	149
5.3 Solutions of a Continuous-Time System: System Responses	152
5.3.1 Some Important Properties of the Matrix e^{At}	156
5.3.2 Sensitivity of e^{At}	157
5.3.3 Computational Methods for e^{At}	158
5.3.4 Comparison of Different Methods for Computing the Exponential Matrix	166
5.3.5 Evaluating an Integral with the Exponential Matrix	167
5.4 State-Space Solution of the Discrete-Time System	168
5.5 Transfer Function and Frequency Response	170
5.5.1 Transfer Function	170
5.5.2 The Frequency Response Matrix and its Computation	172
5.6 Some Selected Software	176
5.6.1 MATLAB CONTROL SYSTEM TOOLBOX	176
5.6.2 MATCONTROL	176
5.6.3 SLICOT	176

5.6.4	MATRIX_X	177
5.7	Summary and Review	178
5.8	Chapter Notes and Further Reading	180
6	CONTROLLABILITY, OBSERVABILITY AND DISTANCE TO UNCONTROLLABILITY	191
6.1	Introduction	192
6.2	Controllability: Definitions and Basic Results	193
6.2.1	Controllability of a Continuous-Time System	193
6.2.2	Controllability of a Discrete-Time System	197
6.3	Observability: Definitions and Basic Results	198
6.3.1	Observability of a Continuous-time System	198
6.3.2	Observability of a Discrete-Time System	200
6.4	Decompositions of Uncontrollable and Unobservable Systems.	200
6.5	Controller and Observer Canonical Forms	202
6.6	Numerical Difficulties with Theoretical Criteria of Controllability and Observability	204
6.7	A Numerically Effective Test of Controllability	207
6.8	A Numerically Effective Test of Observability	216
6.9	Distance to an Uncontrollable System	216
6.9.1	Newton's and the Bisection Methods for Computing the Distance to Uncontrollability	218
6.9.2	The Wicks-DeCarlo Method for Distance to Uncontrollability	222
6.9.3	A Global Minimum Search Algorithm.	224
6.10	Distance to Uncontrollability and the Singular Values of the Controllability Matrix	224
6.11	Some Selected Software	225
6.11.1	MATLAB CONTROL SYSTEM TOOLBOX	225
6.11.2	MATCONTROL	225
6.11.3	CSP-ANM	225
6.11.4	SLICOT	226
6.11.5	MATRIX_X	226
6.12	Summary and Review	226
6.13	Chapter Notes and Further Reading	228
7	STABILITY, INERTIA, AND ROBUST STABILITY	237
7.1	Introduction	238
7.2	Stability of a Continuous-time System	239
7.2.1	Eigenvalue Criterion of Continuous-Time Stability	240
7.2.2	Continuous-time Lyapunov Stability Theory	241
7.2.3	Lyapunov Equations and Controllability and Observability Grammians	246
7.2.4	Lyapunov Equations and the H_2 -Norm	247
7.3	Stability of a Discrete-time System	249

7.3.1	Stability of a Homogeneous Discrete-Time System	249
7.4	Some Inertia Theorems	252
7.4.1	The Sylvester Law of Inertia	252
7.4.2	The Lyapunov Inertia Theorems	253
7.5	Determining the Stability and Inertia of a Nonsymmetric Matrix	256
7.6	Distance to an Unstable System	260
7.7	Robust Stability	266
7.8	The Structured Stability Radius	271
7.9	Some Selected Software	275
7.9.1	MATLAB CONTROL SYSTEM TOOLBOX	275
7.9.2	MATCONTROL	275
7.9.3	SLICOT	275
7.10	Summary and Review	275
7.11	Chapter Notes and Further Reading	277
8	NUMERICAL SOLUTIONS AND CONDITIONING OF LYAPUNOV AND SYLVESTER EQUATIONS	289
8.1	Introduction	291
8.2	The Existence and Uniqueness of Solutions	292
8.2.1	The Sylvester Equation: $XA + BX = C$	292
8.2.2	The Lyapunov Equation: $XA + A^T X = C$	293
8.2.3	The Discrete Lyapunov Equation: $A^T X A - X = C$	293
8.3	Perturbation Analysis and the Condition Numbers	294
8.3.1	Perturbation Analysis for the Sylvester Equation	294
8.3.2	The Condition Number of the Sylvester Equation	296
8.3.3	Perturbation Analysis for the Lyapunov Equation	297
8.3.4	The Condition Number of the Lyapunov Equation	297
8.3.5	Sensitivity of the Stable Lyapunov Equation	298
8.3.6	Sensitivity of the Discrete Lyapunov Equation	301
8.3.7	Sensitivity of the Stable Discrete Lyapunov Equation	301
8.3.8	Determining Ill-Conditioning from the Eigenvalues	302
8.3.9	A Condition Number Estimator for the Sylvester Equation: $A^T X - XB = C$	304
8.4	Analytical Methods for the Lyapunov Equations: Explicit Expressions for Solutions	306
8.5	Numerical Methods for the Lyapunov and Sylvester Equations.	307
8.5.1	Numerical Instability of Diagonalization, Jordan Canonical Form, and Companion Form Techniques.	308
8.5.2	The Schur Method for the Lyapunov Equation: $XA + A^T X = C$	309
8.5.3	The Hessenberg-Schur Method for the Sylvester Equation	313
8.5.4	The Schur Method for the Discrete Lyapunov Equation	318
8.5.5	Residual and Backward Error in the Schur and Hessenberg-Schur Algorithms	321

8.5.6 A Hessenberg Method for the Sylvester Equation: $AX + XB = C$	323
8.5.7 The Hessenberg-Schur Method for the Discrete Sylvester Equation	327
8.6 Direct Computations of the Cholesky Factors of Symmetric Positive Definite Solutions of Lyapunov Equations	328
8.6.1 Computing the Cholesky Factor of the Positive Definite Solution of the Lyapunov Equation	328
8.6.2 Computing the Cholesky Factor of the Positive Definite Solution of the Discrete Lyapunov Equation	333
8.7 Comparisons of Different Methods and Conclusions	336
8.8 Some Selected Software	337
8.8.1 MATLAB CONTROL SYSTEM TOOLBOX	337
8.8.2 MATCONTROL	337
8.8.3 CSP-ANM	337
8.8.4 SLICOT	338
8.8.5 MATRIX _X	338
8.8.6 LAPACK	339
8.9 Summary and Review	339
8.10 Chapter Notes and Further Reading	340
9 REALIZATION AND SUBSPACE IDENTIFICATION	353
9.1 Introduction	354
9.2 State-Space Realizations of a Transfer Function	355
9.2.1 Controllable and Observable Realizations	355
9.2.2 Minimal Realization	357
9.3 Computing Minimal Realizations From Markov Parameters	360
9.3.1 Some Basic Properties of the Hankel Matrix of Markov Parameters	361
9.3.2 An SVD Method for Minimal Realization	362
9.3.3 A Modified SVD Method for Minimal Realization	365
9.4 Subspace Identification Algorithms	370
9.4.1 A Subspace Deterministic Model Identification Algorithm	370
9.4.2 A Stochastic Subspace Model Identification Algorithm	375
9.4.3 Continuous-time System Identification	377
9.4.4 Frequency-Domain Identification	378
9.5 Some Selected Software	380
9.5.1 MATLAB CONTROL SYSTEM TOOLBOX	380
9.5.2 MATCONTROL	380
9.5.3 CSP-ANM	380
9.5.4 SLICOT	380
9.5.5 MATRIX _X	381
9.6 Summary and Review	381
9.7 Chapter Notes and Further Reading	382

10 FEEDBACK STABILIZATION, EIGENVALUE ASSIGNMENT, AND OPTIMAL CONTROL	391
10.1 Introduction	392
10.2 State-Feedback Stabilization	393
10.2.1 Stabilizability and Controllability	394
10.2.2 Stabilization via Lyapunov Equations	396
10.3 Detectability	402
10.4 The Eigenvalue and Eigenstructure Assignment Problems	403
10.4.1 Eigenvalue Assignment by State Feedback	406
10.4.2 Eigenvalue Assignment by Output Feedback	409
10.4.3 Eigenstructure Assignment	410
10.5 The Quadratic Optimization Problems	411
10.5.1 The Continuous-time Linear Quadratic Regulator (LQR) Problem	412
10.5.2 The Discrete-time Linear Quadratic Regulator Problem	420
10.6 H_∞ Control Problems	421
10.6.1 Computing the H_∞ -Norm	423
10.6.2 H_∞ Control Problem: A State Feedback Case.	428
10.6.3 The H_∞ Control Problem: Output Feedback Case	430
10.7 The Complex Stability Radius and Riccati Equation	433
10.8 Some Selected Software	438
10.8.1 MATLAB CONTROL SYSTEM TOOLBOX	438
10.8.2 MATCONTROL	438
10.8.3 CSP-ANM	438
10.8.4 SLICOT	438
10.8.5 MATRIX _X	439
10.9 Summary and Review	439
10.10 Chapter Notes and Further Reading	444
11 NUMERICAL METHODS AND CONDITIONING OF THE EIGENVALUE ASSIGNMENT PROBLEMS	455
11.1 Introduction	456
11.2 Numerical Methods for the Single-input Eigenvalue Assignment Problem	458
11.2.1 A Recursive Algorithm for the Single-input EVA Problem	461
11.2.2 An Error Analysis of the Recursive Single-Input Method	466
11.2.3 The QR and RQ Implementations of Algorithm 11.2.1	467
11.2.4 Explicit and Implicit RQ Algorithms	473
11.3 Numerical Methods for the Multi-input Eigenvalue Assignment Problem	473
11.3.1 A Recursive Multi-input Eigenvalue Assignment Algorithm	474
11.3.2 The Explicit QR Algorithm for the Multi-input EVA Problem	477
11.3.3 The Schur Method for the Multi-input Eigenvalue Assignment Problem .	484
11.4 Conditioning of the Feedback Problem	493

11.4.1	The Single-Input Case	493
11.4.2	The Multi-input Case	496
11.4.3	Absolute and Relative Condition Numbers	497
11.5	Conditioning of the Closed-loop Eigenvalues	499
11.6	Robust Eigenvalue Assignment	501
11.6.1	Measures of Sensitivity	501
11.6.2	Statement of the Robust Eigenvalue Assignment Problem	502
11.6.3	A Solution Technique for the Robust Eigenvalue Assignment Problem . .	503
11.7	Table of Comparison for Efficiency and Stability: Single-input EVA Problem . .	509
11.8	Table of Comparison for Efficiency and Stability: Multi-input EVA Problem . .	510
11.9	Comparative Discussion of Various Methods and Recommendation	511
11.10	Some Selected Software	512
11.10.1	MATLAB CONTROL SYSTEM TOOLBOX	512
11.10.2	MATCONTROL	512
11.10.3	CSP-ANM	512
11.10.4	SLICOT	513
11.10.5	MATRIX _X	513
11.10.6	POLEPACK	513
11.11	Summary and Review	513
11.12	Chapter Notes and Further Reading	516
12	STATE ESTIMATION: Observer and the Kalman Filter	529
12.1	Introduction	530
12.2	State Estimation via Eigenvalue Assignment	531
12.3	State Estimation via Sylvester Equation	532
12.4	Reduced-order State Estimation	535
12.4.1	Reduced-order State Estimation via Eigenvalue Assignment	535
12.4.2	Reduced-order State Estimation via Sylvester-observer Equation	540
12.5	Combined State Feedback and Observer Design	542
12.6	Characterization of Nonsingular Solutions of the Sylvester Equation	544
12.7	Numerical Solutions of the Sylvester-Observer Equation	546
12.7.1	A Recursive Method for the Hessenberg Sylvester-Observer Equation. . .	547
12.7.2	A Recursive Block-Triangular Algorithm for the Hessenberg Sylvester-Observer Equation	551
12.8	Numerical Solution of a Constrained Sylvester-observer Equation	556
12.9	Optimal State Estimation: The Kalman Filter	560
12.10	The Linear Quadratic Gaussian Problem	565
12.11	Some Selected Software	570
12.11.1	MATLAB CONTROL SYSTEM TOOLBOX	570
12.11.2	MATCONTROL	570
12.11.3	CSP-ANM	570

12.11.4 SLICOT	570
12.11.5 MATRIX _X	570
12.12 Summary and Review	571
12.13 Chapter Notes and Further Reading	574
13 NUMERICAL SOLUTIONS AND CONDITIONING OF ALGEBRAIC RICCATI EQUATIONS	583
13.1 Introduction	585
13.2 The Existence and Uniqueness of the Stabilizing Solution of the CARE	587
13.3 The Existence and Uniqueness of the Stabilizing Solution of the DARE	595
13.4 Conditioning of the Riccati Equations	596
13.4.1 Conditioning of the CARE	596
13.4.2 Conditioning of the DARE	601
13.5 Computational Methods for Riccati Equations	604
13.5.1 The Invariant Subspace Methods	605
13.5.2 The Deflating Subspace Methods	614
13.5.3 The Matrix Sign Function Methods	624
13.5.4 Newton's Methods	631
13.6 The Schur and Inverse-Free Generalized Schur Methods for the Descriptor Riccati Equations	645
13.6.1 The Generalized Schur Method for the DCARE	645
13.6.2 The Inverse-Free Generalized Schur Method for the DCARE	646
13.6.3 The Inverse-Free Generalized Schur Method for the DDARE	646
13.7 Conclusions and Table of Comparisons	647
13.8 Some Selected Software	649
13.8.1 MATLAB CONTROL SYSTEM TOOLBOX	649
13.8.2 MATCONTROL	649
13.8.3 CSP-ANM	649
13.8.4 SLICOT	650
13.8.5 MATRIX _X	650
13.9 Summary and Review	650
13.10 Chapter Notes and Further Reading	654
13.11 Introduction	671
13.12 The Existence and Uniqueness of the Stabilizing Solution of the CARE	673
13.13 The Existence and Uniqueness of the Stabilizing Solution of the DARE	681
13.14 Conditioning of the Riccati Equations	682
13.14.1 Conditioning of the CARE	682
13.14.2 Conditioning of the DARE	687
13.15 Computational Methods for Riccati Equations	690
13.15.1 The Invariant Subspace Methods	691
13.15.2 The Deflating Subspace Methods	700

13.15.3 The Matrix Sign Function Methods	710
13.15.4 Newton's Methods	717
13.16 The Schur and Inverse-Free Generalized Schur Methods for the Descriptor Riccati Equations	731
13.16.1 The Generalized Schur Method for the DCARE	731
13.16.2 The Inverse-Free Generalized Schur Method for the DCARE	732
13.16.3 The Inverse-Free Generalized Schur Method for the DDARE	732
13.17 Conclusions and Table of Comparisons	733
13.18 Some Selected Software	735
13.18.1 MATLAB CONTROL SYSTEM TOOLBOX	735
13.18.2 MATCONTROL	735
13.18.3 CSP-ANM	735
13.18.4 SLICOT	736
13.18.5 MATRIX _X	736
13.19 Summary and Review	736
13.20 Chapter Notes and Further Reading	740
14 INTERNAL BALANCING AND MODEL REDUCTION	757
14.1 Introduction	758
14.2 Internal Balancing for Continuous-time Systems	759
14.2.1 Internal Balancing of a Minimal Realization	759
14.2.2 Internal Balancing of a Nonminimal Realization	764
14.3 Internal Balancing For Discrete-time Systems	766
14.4 Model Reduction	768
14.4.1 Model Reduction via Balanced Truncation	769
14.4.2 The Schur Method for Model Reduction	772
14.4.3 A Balancing-Free Square-Root Method for Model Reduction	780
14.5 Hankel-Norm Approximations	781
14.5.1 A Characterization of All Solutions to Hankel-Norm Approximation	781
14.6 Model Reduction of an Unstable System	790
14.7 Frequency Weighted Model Reduction	791
14.8 Summary and Comparisons of Model Reduction Procedures	792
14.9 Some Selected Software	794
14.9.1 MATLAB CONTROL SYSTEM TOOLBOX	794
14.9.2 MATCONTROL	794
14.9.3 CSP-ANM	794
14.9.4 SLICOT	794
14.9.5 MATRIX _X	794
14.10 Summary and Review	795
14.11 Chapter Notes and Further Reading	797

15 LARGE-SCALE MATRIX COMPUTATIONS IN CONTROL: KRYLOV SUB- SPACE METHODS	809
15.1 Introduction	810
15.2 A General Discussion on Krylov Subspace Methods	811
15.3 The Arnoldi and GMRES Methods	813
15.3.1 The Arnoldi Method	813
15.3.2 Solving $\mathbf{Ax} = \mathbf{b}$ using the Arnoldi Method	815
15.3.3 The GMRES Method for Solving $\mathbf{Ax} = \mathbf{b}$	820
15.3.4 Solving Shifted Linear Systems Using the Arnoldi Method	821
15.3.5 The Block Arnoldi Method	822
15.4 The Lanczos Method	823
15.4.1 The Block Lanczos Method.	825
15.5 The Krylov Subspace Methods in Control	826
15.5.1 Scopes of using the Krylov Subspace Methods in Control	826
15.5.2 The Controllability and the observability problems and the Krylov Sub- space Methods	827
15.5.3 Arnoldi Method For Rank-one Lyapunov Equation	828
15.5.4 Restarted Arnoldi Method for Lyapunov Equation	833
15.5.5 Block Arnoldi Method For Discrete Lyapunov Equation	834
15.5.6 Arnoldi Methods for Sylvester Equation	838
15.5.7 Block Arnoldi Methods for Sylvester Equation	843
15.5.8 Arnoldi-Method for Sylvester-Observer Equation (single-output case)	848
15.5.9 Arnoldi-Method for Continuous-time Algebraic Riccati Equation	854
15.5.10 Arnoldi Method for Partial Eigenvalue Assignment	858
15.5.11 Lanczos and Arnoldi Methods for Model Reduction	862
15.6 Summary and Review	879
15.7 Chapter Notes and Further Reading	883
16 NUMERICAL METHODS FOR MATRIX-SECOND-ORDER CONTROL SYSTEMS	895
16.1 Introduction	896
16.2 First-order Representations	899
16.3 The Quadratic Eigenvalue Problem (QEP)	901
16.3.1 Linearization of the Quadratic Eigenvalue Problem	903
16.3.2 Solving the Quadratic Eigenvalue Problem.	905
16.3.3 Computation of the Partial Spectrum of the Quadratic Eigenvalue Prob- lem: Shift and Invert Strategy and the Jacobi-Davidson Method	906
16.4 Independent Modal Control (IMSC) Approach	909
16.4.1 Modal Solution of the State Feedback Problem	911
16.4.2 Modal Solution of the Output Feedback Problem	911
16.4.3 Engineering and Computational Difficulties with the IMSC Approach	912

16.4.4 Simultaneous Diagonalization of the Pencil $K - \lambda M$	913
16.4.5 Simultaneous Triangularization Under Modal Damping	914
16.5 Controllability, Observability, and Stability of Second-order Systems	915
16.5.1 Introduction	915
16.5.2 Eigenvalue Criteria of Controllability and Stabilizability	916
16.5.3 Stability of Second-order systems	918
16.6 Eigenvalue Bounds, Orthogonality of the Eigenvectors, and the Rayleigh Quo- tient of Quadratic Matrix Pencil	925
16.6.1 Eigenvalue Bounds for the Quadratic Pencil	925
16.6.2 Orthogonality of the Eigenvectors of the Quadratic Pencil	927
16.6.3 Rayleigh-Quotient Expressions for Quadratic Pencil	930
16.7 Nonmodal and Partial Modal Approaches to Stabilization, Partial Eigenvalue and Eigenstructure Assignments Using Feedback Control	933
16.7.1 Introduction	933
16.7.2 Problem Statements	934
16.7.3 A Nonmodal Solution of the Feedback Stabilization Problem	937
16.7.4 A Direct and Partial Modal Approach for Partial Eigenvalue and Eigen- structure Assignment	939
16.7.5 Robust Eigenvalue Assignment in Second-order System	952
16.8 Summary and Review	962
16.9 Chapter Notes and Further Reading	967
A SOME EXISTING SOFTWARE FOR CONTROL SYSTEMS DESIGN AND ANALYSIS	979
A.1 MATLAB CONTROL SYSTEM TOOLBOX	980
A.2 MATCONTROL	980
A.3 CONTROL SYSTEM PROFESSIONAL – ADVANCED NUMERICAL METH- ODS (CSP-ANM).	980
A.4 SLICOT	981
A.5 MATRIX _X	981
A.6 SYSTEM IDENTIFICATION SOFTWARE	982
A.6.1 MATLAB System Identification Toolbox	982
A.6.2 Xmath Interactive System Identification Module, Part-2	982
A.6.3 ADAPT _X	982
B MATCONTROL AND LISTING OF MATCONTROL FILES	985

Preface

Remarkable progress has been made in both theory and applications of all important areas of control theory. Theory is rich and sophisticated. Some beautiful applications of control theory are presently being made in aerospace, biomedical engineering, industrial engineering, robotics, economics, power systems, etc. Unfortunately, the same assessment of progress does not hold in general for computations in control theory.

Many of the methods described in earlier control and systems theory text books were developed before the computer era and were based on approaches that are not numerically sound. Most of these methods, for example, require reduction of the system matrices to some condensed formula, such as a companion form or the Jordan canonical form, and it is well-known that these forms cannot, in general, be achieved in a numerically stable way.

The situation is, however, changing quite fast. In the last twenty years or so, numerically viable algorithms have been developed for many of the common linear control problems. Softwares based on these methods have been developed and are still being built.

Unfortunately, these methods and softwares do not seem to be widely known and easily accessible to broad groups of applied mathematicians, control theorists, and practicing control engineers. They are still largely confined in reprints and preprints (in this context it is noted that a reprint book on "*Numerical Linear Algebra Techniques for Systems and Control*" edited by R.V. Patel, A. Laub, and P. Vandooren containing a large number of important published papers in this area has recently been published by IEEE/CRC Press). The primary reason for the inaccessibility of these algorithms and the softwares, in my opinion, is that an understanding, efficient implementations, and making possible modifications of these methods needed for some applications of special interests, require an interdisciplinary knowledge of linear algebra, numerical linear algebra, control theory, and computer science; and such a combined expertise is hard to find.

What is, therefore, needed is a book that makes these algorithms accessible to a wide variety of users, researchers, and students.

For practicing users, it is important that the algorithms are described in a manner that is suitable for easy implementation on a wide range of computers, that important aspects of implementations are discussed, and a clear comparative study of one algorithm over the other for a given problem with respect to efficiency, storage, numerical stability, etc., is presented. The later will help the users to choose the one most suitable for his or her applications. Furthermore,

for the students and researchers, it is important that the mechanism of the development of the algorithms is clearly explained and aspects of perturbation analysis of the problems and round-off error analyses and convergence properties of the algorithms, whenever available, are included in some details.

Of course, all these need to be accomplished requiring a minimal amount of background in the areas mentioned above. This is certainly a difficult and an ambitious task. *But the present book aspires to do that and aims at reaching out to a broad spectrum of audience in a number of disciplines including mathematics, control and systems engineering, and other applications areas such as vibrations, aerospace and structural engineering.*

The recent book on “Computational Methods for Linear Control Systems” by P. H. Petkov, N.D. Christov, and M. M. Konstantinov also aims to fulfill that need to some extent. The scope of this book is, however, much more limited than that of the present book.

The current book is an outgrowth of lecture notes compiled by the author over several years for a graduate course in *numerical methods in control theory* taught at Northern Illinois University (almost all students of this course have been mathematics students with no prior background in control theory). The book has also been used in several short courses given by the author including the SIAM short course on *Numerical Methods in Control, Signal, and Image Processing*, Seattle, August 15, 1993 and, the short course on “*Numerical Methods for Linear Control and Systems*” at the *International Conference on Mathematical Theory of Networks and Systems*, St. Louis, 1996. The audience of these short courses had varying backgrounds.

The book covers most important and relevant problems arising in control system design and analysis with a special emphasis on computational aspects. These include:

- Numerical solutions of state equations and frequency response computations
- Controllability, observability and distance to controllability
- Stability, inertia, robust stability and distance to instability
- Numerical solutions and conditioning of Lyapunov, Sylvester and algebraic Riccati equations
- Numerical algorithms for feedback stabilization, eigenvalue and robust eigenvalue assignment and conditioning of the eigenvalue assignment problem
- Numerical algorithms for full-order and reduced-order observer design and Kalman filtering
- Realization and subspace algorithms for model identification
- Algorithms for balanced realization and model reduction
- Large-scale solutions of control problems
- Effective numerical methods for feedback control in matrix second-order systems

- H_2 and H_∞ control

The numerical algorithms described in the book have the following desirable features:

- **Efficiency.** Algorithms are of order $O(n^3)$.
- **Numerical Stability.** Algorithms are either numerically stable or composed of numerically stable computations.
- **State-of-the-art Algorithms.** The state-of-the-art algorithms for all problems have been included.
- **Comparative Study and Recommendations.** Whenever possible, a comparison of various algorithms for the same problem with respect to efficiency, numerical stability and accuracy has been given and based on this comparative study, recommendation for practicing engineers has been made.
- **Step by Step Explanation.** All algorithms have been explained step by step with illustrative examples illustrating each step of the algorithm.
- **Software and Implementations.** Important selected software for each topic has been included.
- **MATLAB Toolkit.** There exists a MATLAB toolkit called **MATCONTROL**, implementing major algorithms in the book.
- **Algorithms for both Continuous-time and Discrete-time systems.** Algorithms are described both for continuous-time and discrete-time systems.
- **Algorithms for Large-scale Control Problems in both First-order and Matrix Second-order Systems.** This is the first book containing algorithms suitable for large-scale solutions of control problems in standard first-order system, as well as practically implementable and computationally feasible algorithms for feedback control in matrix second-order systems.

The discussions on theoretical aspects of control theory have been kept to a minimum, only the relevant facts have been mentioned. However, the importance and applications of the problems have been discussed to an extent to motivate the readers in mathematics and other areas of science and engineering who are not familiar with control problems. Numerical Linear Algebra techniques needed to understand and implement the algorithms have been developed in the book itself in a concise manner without going into too much details and attempts have been made to make the techniques understandable to the readers who do not have a prior background in numerical linear algebra and numerical analysis. Of course, people having a background in numerical analysis or numerical algebra and/or control theory will have a definite advantage.

A special emphasis has been given to the clear understanding of the distinction between a ‘bad’ algorithm and a ‘numerically effective’ algorithm.

Some discussions on *large-scale computing in control* have been included too. The research in this area is still in its infancy, but some aspects of current research have been included to give the readers a flavor. There is an urgent need for an expanded research in this area as outlined in the 1988 NSF panel report: “**Future Directions in Control Theory: A Mathematical Perspective**”. It is hoped our short coverage in this area will provide enough incentive and motivation to beginning researchers, both from control theory and applied and computational mathematics, to work in the area.

The MATLAB toolkit *MATCONTROL* will help the students and the users understand the merits and drawbacks of one algorithm over the others and possibly help a user to make a right decision in choosing an ideal algorithm for a particular application.

Organization of the Book:

The book has **sixteen** chapters. These sixteen chapters have been organized into **four parts**; each part consisting of several chapters, grouped together (roughly) with a common theme.

Part I. REVIEW OF LINEAR AND NUMERICAL LINEAR ALGEBRA

- Chapter 2. A Review of Some Basic Concepts and Results from Theoretical Linear Algebra**
- Chapter 3. Some Fundamental Tools and Concepts from Numerical Linear Algebra**
- Chapter 4. Canonical Forms Obtained via Orthogonal Transformations**

Part II. CONTROL SYSTEM ANALYSIS

- Chapter 5. Linear State Space Models and Solutions of the State Equations**
- Chapter 6. Controllability, Observability and Distance to Uncontrollability**
- Chapter 7. Stability, Inertia, and Robust Stability**
- Chapter 8. Numerical Solutions and Conditioning of Lyapunov and Sylvester Equations**

Part III. CONTROL SYSTEMS DESIGN

- Chapter 9. Realization and Subspace Identification**
- Chapter 10. Feedback Stabilization, Eigenvalue Assignment, and Optimal Control**

- Chapter 11. Numerical Methods and Conditioning of the Eigenvalue Assignment Problems**
- Chapter 12. State Estimation: Observer and the Kalman Filter**
- Chapter 13. Numerical Solutions and Conditioning of Algebraic Riccati Equations**
- Chapter 14. Internal Balancing and Model Reduction**

Part IV. SPECIAL TOPICS

- Chapter 15. Large-Scale Matrix Computations in Control: Krylov Subspace Methods**
- Chapter 16. Numerical Method for Matrix-Second-Order Control Systems**

The book can be used as a textbook for an advanced graduate course in control engineering such as “*Computational Methods for Control Systems Design and Analysis*” and “*Computer-aided Control System Design*” or for an advanced graduate topic course on “*Numerical Linear Algebra Techniques in Control and Systems*” in applied mathematics and scientific computing. Far more material than can be covered in one semester has been included, so professors can tailor material to particular courses and develop course syllabi. *Above all, the book is intended to serve as a reference book for practicing engineers and applied scientists, researchers and graduate students.* The book is also very suitable for **self-study**.

Acknowledgments

This book would not exist in its present form without the help, guidance, suggestions and encouragement of many friends and colleagues around the world.

My deepest gratitude is to the following people, who not only read and re-read several Chapters carefully and diligently but actually made valuable contributions. They are: Professor **Kanti Datta** of Indian Institute of Technology, Kharagpur, India; Dr. **Vasile Sima** of Research Institute of Informatics, Bucharest, Romania, and two of my former Ph.D. students, **João Carvalho** (now at Universidade Federal de Rio Grande do Sol, Porto Alegre, Brazil), and **Daniil Sarkissian** (now at Mississippi State University). Drs. Carvalho and Sarkissian also wrote the M-files for the MATLAB-based software **MATCONTROL** that accompanies the book.

The following people read certain chapters carefully and thoroughly, pointed out errors, and gave comments and suggestions for improvements: Professors **Thanos Antoulas** of Rice University; **Mark Arnold** of University of Arkansas; **Daniel Boley** of University of Minnesota; **Ralph Byers** of University of Kansas; **Peter Benner** of Technical University of Hamburg, Germany; **James Bunch** of University of California; **Raghu Balakrishnan** of Purdue University;

Chapter 1

INTRODUCTION AND OVERVIEW

Contents

1.1	Linear and Numerical Linear Algebra (Chapter 2 and Chapters 3 and 4)	4
1.2	System Responses (Chapter 5)	7
1.3	Controllability and Observability problems (Chapter 6)	9
1.4	Stability and Inertia (Chapter 7)	10
1.5	Lyapunov, Sylvester, and Algebraic Riccati Equations (Chapter 8 and Chapter 13)	11
1.6	Realization and Identification (Chapter 9)	14
1.7	Feedback Stabilization and Eigenvalue Assignment (Chapter 10 and Chapter 11)	15
1.8	State Estimation (Chapter 12)	17
1.9	Internal Balancing and Model Reduction (Chapter 14)	19
1.10	Krylov Subspace Methods for Large and Sparse Control Problems (Chapter 15)	20
1.11	Numerical Methods for Control Problems Modeled by Systems of Second-Order Differential Equations (Chapter 16)	20
1.12	Nearness to Uncontrollability and Instability (Chapter 6 and Chapter 7) and Robust Stability and Stability Radius (Chapter 7 and Chapter 10)	22
1.12.1	Nearness to Uncontrollability and Instability	22
1.12.2	Robust stability and stability Radius (Chapter 7 and Chapter 10)	23
1.13	Sensitivity and Condition Numbers of Control Problems	24
1.13.1	Conditioning of the Matrix Exponential Problem (Chapter 5)	25
1.13.2	Conditioning of the Lyapunov and Sylvester Equations (Chapter 8)	25

1.13.3 Conditioning of the Algebraic Riccati Equations (Chapter 13)	26
1.13.4 Conditioning of the Feedback and Eigenvalue Assignment Problems (Chapter 11)	27
1.14 H_∞-Control (Chapter 10)	27
1.15 Software for Control Problems	29

A linear time-invariant **continuous-time dynamical system in state-space** is described by the matrix differential equations of the form

$$\dot{x}(t) = Ax(t) + Bu(t); \quad x(t_0) = x_0, \quad t \geq t_0 \quad (1.0.1)$$

$$y(t) = Cx(t) + Du(t), \quad (1.0.2)$$

where A , B , C , and D are real time-invariant $n \times n$ state matrix, $n \times m$ ($m \leq n$) input matrix, $r \times n$ ($r \leq n$) output matrix, and $m \times r$ direct transmission matrix, respectively. The vectors u , x , and y are time-dependent vectors referred to as *input*, *state*, and *output*, respectively. The dot, ‘ \dot{x} ’, denotes ordinary differentiation with respect to t . If $m = 1$, then the matrix B is an $n \times 1$ column vector, and is denoted by b . The control problem dealing with such an input vector is referred to as the *single-input problem* (because u is a scalar in this case). The *single-output* problem is analogously defined.

Similarly, a **linear time-invariant discrete-time dynamical system in state-space** is represented by the vector-matrix difference equations of the form

$$x(k+1) = Ax(k) + Bu(k); \quad x(0) = x_0, \quad k \geq 0 \quad (1.0.3)$$

$$y(k) = Cx(k) + Du(k). \quad (1.0.4)$$

For notational convenience, the system (1.0.1)-(1.0.2) or its discrete counterpart (1.0.3)-(1.0.4) is sometimes denoted simply by (A, B, C, D) . **The matrix D will be assumed to be a zero matrix for most problems in this book.**

The transfer function matrix from u to y for the system (1.0.1)-(1.0.2) is defined as

$$\hat{y}(s) = G(s)\hat{u}(s)$$

where $\hat{u}(s)$ and $\hat{y}(s)$ are the Laplace transforms of $u(t)$ and $y(t)$ with $x(0) = 0$. Thus

$$G(s) = C(sI - A)^{-1}B + D.$$

Sometimes the notation

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right] = C(sI - A)^{-1}B + D$$

will be used for simplicity.

The transfer function matrix for a discrete-time system is similarly defined.

This book deals with computational methods for control problems modeled by the systems of the above types; and with numerical analysis aspects associated with these computational methods, such as, conditioning of problems, numerical stability of algorithms, accuracy of solutions, etc. Furthermore, numerically effective computational methods for feedback control in matrix second-order systems, which can be implemented without first-order linearization to systems of the form (1.0.1), have been described.

The following major topics, associated with the design and analysis of linear control system have been addressed in the book:

- Linear and Numerical Linear Algebra
- System Responses
- Controllability and Observability
- Stability and Inertia
- Lyapunov, Sylvester and Riccati Equations
- Realization and Identification
- Feedback Stabilization and Eigenvalue Assignment
- State Estimation
- Internal Balancing and Model Reduction
- Krylov Subspace Methods for Large and Sparse Control Problems
- Numerical Methods for Control Problems Modeled by Systems of Matrix Second-order Differential Equations
- Nearness to Uncontrollability and Instability
- Sensitivity and Condition Numbers for Eigenvalue Assignment; Lyapunov, Sylvester and Riccati equations
- H_2 and H_∞ Control
- Selected Control Software

In the sequel, we now give an overview of each of these topics with references to the Chapter(s) and Section(s) in which it is dealt with. *For references of the papers cited in these sections, please consult the reference sections of the associated chapters.*

1.1 Linear and Numerical Linear Algebra (Chapter 2 and Chapters 3 and 4)

The linear and numerical linear algebra background needed to understand the computational methods has been done in the book itself in **Chapters 2-4**.

All major aspects of numerical matrix computations including solutions and least-squares solutions of algebraic linear systems, eigenvalue and singular value computations, computations of generalized eigenvalues and eigenvectors, etc., have been covered. The following methods are numerically reliable and widely used in practice:

- Gaussian elimination method with partial pivoting for LU factorization and linear algebraic systems
- Householder's and Givens' methods for QR factorization
- Householder's and Givens' methods for reduction to Hessenberg forms; and the QR and QZ Iteration methods, for reductions to real Schur and generalized real Schur forms, respectively.
- The QR factorization and the SVD methods for least-squares solutions
- The QR iteration method with implicit double shift for eigenvalue computations
- The QZ iteration method for generalized eigenvalues computations
- The SVD method for computing numerical rank and nearness to rank deficiency and for finding the generalized inverse.

The basic but most important concepts in numerical linear algebra; namely, the concepts of the **conditioning of a problem**, the **numerical stability** of an algorithm and their effects on the accuracy of the computed solution have been briefly but clearly explained.

The most important thing here is to note that the *accuracy of the computed solution of a problem by a given algorithm is directly affected both by the conditioning of the problem and the numerical stability of the algorithm*. If a problem is ill-conditioned, no matter what algorithm is used, the accuracy of the solution cannot be guaranteed.

An algorithm is *backward stable* if the computed solution is the exact solution of a nearby problem. The backward stability (henceforth referred to as numerical stability or simply as stability of an algorithm) is determined by round-off error analysis.

The conditioning or the sensitivity of a problem is determined by means of a number, called the **condition number**. If the condition number is large, the problem is called **ill-conditioned**; otherwise, it is **well-conditioned**.

- The condition number of a matrix (and of the problem $Ax = b$) is $\| A \| \| A^{-1} \|$, in case A is square and nonsingular; and is $\| A \| \| A^\dagger \|$, in case A is nonsquare or singular. The condition number of A with respect to 2-norm is $Cond_2(A) = \frac{\sigma_1}{\sigma_n}$, where σ_1 and σ_n are, respectively, the largest and the smallest singular values of A .
- The condition number of an orthogonal matrix O with respect to 2-norm is 1:

$$Cond_2(O) = 1.$$

- The condition number of a simple eigenvalue λ of A is defined as $s = |y^*x|$, where x and y are right and left eigenvectors of A associated with λ , respectively.

The overall sensitivity of the eigenvalues of a matrix depends upon the condition number of the eigenvector matrix (**Bauer-Fike Theorem (Theorem 3.3.3)**).

- The eigenvalues of a symmetric matrix and the singular values of a matrix are well-conditioned.

Another important aspect of a numerical algorithm is its *efficiency*. A matrix algorithm involving computations with $n \times n$ matrices is said to be *efficient* if it does not require more than $O(n^3)$ floating point operations (*flops*). *Note that an algorithm may be efficient but unstable.* The Gaussian elimination algorithm without pivoting is efficient, but is unstable in general.

Canonical Forms

A common strategy for numerically solving control problems can be described in the following steps taken in the sequence:

- Step 1.** The problem is transformed by reducing the matrices A , B , and C to some convenient “condensed” forms using transformations that preserve the desirable properties of the problem at hand.
- Step 2.** The transformed problem is solved by exploiting the structure of the condensed forms of the matrices A , B , and C obtained in Step 1.
- Step 3.** The solution of the original problem is recovered from the solution of the transformed problem.

Extreme caution should be taken in implementing Step 1 in floating point computations. Of course, Steps 2 and 3 must be implemented also in a numerically stable way, but things might go wrong in the first place. We show in the following how it happens in case of a similarly transformation.

Let $fl(s)$ denote the floating point computation of a quantity s , and let the matrix A be transformed by similarity to some condensed form. Then it can be shown [see Golub and Van Loan (1996), page 317], that:

$$fl(X^{-1}AX) = X^{-1}AX + E, \text{ where } \|E\|_2 \approx \mu \text{Cond}_2(X) \|A\|_2$$

(μ is the machine-precision). Since the error matrix E clearly depends upon $\text{Cond}_2(X)$, it follows that if X is ill-conditioned, then the matrix $X^{-1}AX$ cannot be computed accurately.

Two condensed forms that have been used often in the past in control literature are: the *Jordan Canonical Form* (JCF) and the *Frobenius* (or *Block Companion*) Form (a variation of this is known as the *Luenberger Canonical Form*). Exploitation of rich structures of these forms often makes it much easier to solve a problem and these forms are very convenient for textbook illustrations.

Unfortunately, determination of both these forms might require very ill-conditioned similarity transformations.

For the Jordan canonical form, the transforming matrix X is ill-conditioned if the matrix A is defective or nearly defective.

For a companion form, the process of reduction comes in two stages: First, A is transformed to a Hessenberg matrix H , and second, the transformed (unreduced) Hessenberg matrix H is further reduced to a companion matrix. *The first stage can be achieved in a numerically stable way* using Householder's or Givens' method [see Datta (1995)] with an orthogonal similarity transformation (note that an orthogonal matrix is well-conditioned); however, *the second stage can be highly unstable*. The transforming matrix that transforms the Hessenberg matrix H further to a companion matrix will be severely ill-conditioned if the product of the entries on the subdiagonal of the upper Hessenberg matrix H is small.

Suggestions. *Avoid the use of the JCF and companion canonical forms in numerical computations, and use only canonical forms that can be obtained using well-conditioned transforming matrices, such as orthogonal transformations. The Hessenberg form, the controller-Hessenberg and the observer-Hessenberg forms, the real Schur and the generalized real Schur forms, the Hessenberg-triangular form, are examples of such canonical forms. These forms can be obtained via orthogonal transformations.*

Orthogonal transformations use orthogonal matrices and orthogonal matrices have several remarkable features that make them so attractive for numerical computations. They include:

- (i) the inverse of an orthogonal matrix is just its transpose,
- (ii) an orthogonal matrix is perfectly conditioned and
- (iii) the 2- and Frobenius norms of an orthogonal matrix A remain invariant under orthogonal matrix multiplications.

Thus, *the errors in numerical computations involving orthogonal matrix multiplications are not magnified by the process and the sensitivity of a computational problem remains unaffected by the use of orthogonal transformations.*

1.2 System Responses (Chapter 5)

For the continuous-time system (1.0.1-1.0.2) the dynamical system responses $x(t)$ and $y(t)$ for $t \geq t_0$ can be determined from the following formulas:

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^t e^{A(t-s)}Bu(s) ds \quad (1.2.1)$$

$$y(t) = Cx(t) + Du(t) \quad (1.2.2)$$

In order to study the behavior of a dynamical system, it is customary to determine the responses of the system due to different inputs. Two most common inputs are the **unit step** function and the **unit impulse**.

Thus, the **unit step response** of a system is the output that occurs when the input is the unit step function (it is assumed that $x(0) = 0$). Similarly, the **unit impulse response** is the output that occurs when the input is the unit impulse.

The **impulse response matrix** of the system (1.0.1) and (1.0.2) is defined by

$$H(t) = Ce^{At}B + D\delta(t)$$

where $\delta(t)$ is the Dirac delta function. The impulse response is the response of the system to a Dirac input $\delta(t)$.

Thus, to obtain different responses, one needs to compute the exponential matrix

$$e^{At} = I + At + \frac{A^2t^2}{2} + \dots$$

and the integrals involving this matrix. *The computational challenge here is how to determine e^{At} without explicitly computing the matrix powers.* Finding higher powers of a matrix is computationally intensive and is a source of instability for the algorithm that requires such computations.

A well-known example of this appears in Moler and Van Loan [1978]. There, it was attempted to compute e^{At} , where

$$A = \begin{pmatrix} -49 & 29 \\ -64 & 31 \end{pmatrix},$$

using the Taylor series:

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}.$$

The obtained output with $k = 59$ was found to be:

$$e^A \approx \begin{pmatrix} -22.25880 & -1.432766 \\ -61.49931 & -3.474280 \end{pmatrix},$$

whereas the correct answer to 6 decimal places is

$$e^A \approx \begin{pmatrix} -0.735759 & 0.551819 \\ -1.471518 & 1.103638 \end{pmatrix}.$$

Another obvious way to compute e^A is to use some simple canonical forms of A such as the *Jordan canonical form* or a *companion form of A*. It is shown in Chapter 5 by simple examples how such computations can lead to inaccurate results.

The method of choice here is either *Padé approximation with scaling and squaring* (**Algorithm 5.3.1**) or the *method based on reduction of A to real Schur form* (**Algorithm 5.3.2**).

Computations of system and impulse responses for the discrete-time system (1.0.3)-(1.0.4) require evaluation of various powers of the matrix A (see **Chapter 5**). The powers of the matrix A , if needed to be computed, should be computed by reducing the matrix A first to a Hessenberg or to a triangular matrix which can be achieved using orthogonal transformations. Finally, a method (**Algorithm 5.3.3**) for computing an integral involving an exponential matrix is described in Section 5.3.3. The method is due to Van Loan (1978).

Frequency Response Computations

The frequency response plot for many different values of the frequency ω is important in the study of various important properties of linear systems. The frequency-response curves indicate how the magnitude and angle of the sinusoidal steady-state response change as the frequency of the input is changed. For this, the frequency response matrix $G(j\omega) = C(j\omega I - A)^{-1}B + D(\omega \geq 0)$ needs to be computed. Computing $G(j\omega)$ using the LU decomposition of A would require $O(n^3)$ operations per ω and is, therefore, not practical when this computation has to be done for a large number of values of ω . An efficient and practical method due to Laub (1981), based on reduction of A to a Hessenberg matrix, is presented in **Algorithm 5.5.1**, and discussions on some other recent methods for efficient computations of the frequency response matrix is included in **Section 5.5.2**.

1.3 Controllability and Observability problems (Chapter 6)

The system (1.0.1) is controllable or, equivalently, the pair (A, B) is controllable, if for any initial state $x(0) = x_0$ and the final state x_f , there exists an input $u(t)$ such that the solution satisfies $x(t_f) = x_f$. Several mathematically equivalent criteria of controllability are stated and proved in **Theorem 6.2.1**; the most well-known of them being the **Kalman criterion of controllability**, which states that the pair (A, B) is controllable if and only if the controllability matrix

$$C_M = (B, AB, \dots, A^{n-1}B) \quad (1.3.1)$$

has full rank.

The following simple example due to Page (1981):

$$A = \text{diag}(2^0, 2^{-1}, 2^{-2}, \dots, 2^{-9}); \quad B = (1, 1, \dots, 1)^T,$$

shows that *the Kalman criterion does not yield to a numerically viable test for controllability*. For this pair, the controllability matrix C_M can be easily computed and stored; but it has three small singular values 0.613×10^{-12} , 0.364×10^{-4} , and 0.712×10^{-7} showing that C_M is rank deficient in numerical computations with machine precision no smaller than 10^{-12} . This will lead to the conclusion that (A, B) is not controllable, whereas the pair (A, B) is clearly controllable.

Similar remarks hold for other criteria. See **Example 6.6.2** in Chapter 6 which demonstrates the pitfall of the eigenvalue criterion (popularly known as the **Hautus-Popov-Belevich criterion**).

A numerically viable test of controllability, based on the reduction to (A, B) to the controller-Hessenberg form, is given in Section 6.8.

Observability is a dual concept to controllability. Thus, all that we have said above about controllability applies equally to observability.

1.4 Stability and Inertia (Chapter 7)

It is well-known that the uncontrolled system

$$\dot{x} = Ax(t) \quad (1.4.1)$$

is asymptotically stable if and only if all the eigenvalues of A have negative real parts.

Similarly, the discrete system

$$x(k+1) = Ax(k) \quad (1.4.2)$$

is asymptotically stable if and only if the eigenvalues of A have moduli less than 1.

The common approaches for determining the stability of a system include (a) finding the characteristic polynomial of A followed by application of the Routh-Hurwitz test in case of continuous-time stability or the Schur-Cohn criterion in case of discrete-time stability (b) solving and testing the positive definiteness of the solution matrix X of the associated Lyapunov equations:

$$XA + A^T X = -I \text{ (for continuous-time stability)} \quad (1.4.3)$$

or

$$X - A^T X A = I \text{ (for discrete-time stability).} \quad (1.4.4)$$

Finding the characteristic polynomial of a matrix is potentially a numerically unstable process and furthermore, the coefficients of the characteristic polynomial can be extremely sensitive to small perturbations (see Chapter 4). The Lyapunov equation approach is counter-productive in the sense that the most numerically viable method for solving a Lyapunov equation namely, the **Bartels-Stewart** method is based on the reduction of A to a real Schur form, and the latter either explicitly displays the eigenvalues of A or they can be trivially computed.

It is, therefore, commonly believed that the most viable way to test the stability of a system is to compute the eigenvalues of A using the universally-used method, called the *QR* iteration with double shift (see Chapter 4).

Having said this, let's note that with explicit computation of eigenvalues, one gets much more than what is needed for determining the stability, and moreover, as just said, the eigenvalues can be extremely ill-conditioned. *An indirect method that neither explicitly solves a Lyapunov equation nor computes the eigenvalues, was developed by Carlson and Datta (1979).* This method was later modified by Datta and Datta (1981). **According to theoretical operations-count, both these methods are about three to four times faster than the eigenvalue method and many times faster than the Lyapunov equation method.** The Carlson-Datta method is described in **Algorithm 7.5.1**.

Several inertia theorems (**Theorem 7.4.1-7.4.6**) are stated and some are proved in **Section 7.4**.

1.5 Lyapunov, Sylvester, and Algebraic Riccati Equations (Chapter 8 and Chapter 13)

The Lyapunov equations

$$XA + A^T X = C \quad (1.5.1)$$

$$X - A^T X A = C \quad (1.5.2)$$

arise in

- Stability and Robust Stability Analyses (**Chapter 7**)
- Model Reduction (**Chapter 14**)
- Internal Balancing (**Chapter 14**)
- Determining H_2 -norm (**Chapter 7**)

The Sylvester equation

$$XA + BX = C \quad (1.5.3)$$

arises in the design of observer (**Chapter 12**), and it can also be used to solve feedback stabilization and pole-placement problems (**Chapters 10 and 11**)

Solving these equations via reduction of A and/or B to a companion form or the Jordan canonical form is numerically unreliable; because, as said before, these forms cannot be, in general, obtained in a numerically stable way.

Experience with numerical experiments reveal that *solving Lyapunov equations of order higher than twenty using a companion form of A yields solutions with errors as large as the solutions themselves*. **Example 8.5.1** in Chapter 8 illustrates the danger of solving a Lyapunov equation using the Jordan canonical forms of A . With A and C as chosen in **Example 8.5.1**, the *solution of (1.5.1) via diagonalization of A (available in MATLAB function `lyap2`) does not have any single entry common with the exact solution*.

The methods of choice are:

- *The Schur method (**Section 8.5.2**) for the Lyapunov equation*
- *The Hessenberg-Schur method (**Algorithm 8.5.1**) for the Sylvester equation*

The Schur algorithm due to Bartels and Stewart (1972) is based on reduction of A to real Schur form. The Hessenberg Schur algorithm due to Golub, Van Loan and Nash (1979) is based on reduction of the larger of A and B to Hessenberg form and the other to real Schur form.

These methods are based on numerically stable reduction procedures and though only some forms of weak stability (*the algorithms are at most conditionally backward stable*) have been proved, **they are the best available so far and widely used in practice**.

In several applications, all that is needed is the Cholesky factor L of the symmetric positive definite solution X of a Lyapunov equation. For example, the controllability and observability Grammians of a stable system play an important role in **balanced realization** and **model reduction (Chapter 14)**. These Grammians, are, respectively, the positive definite solutions of the Lyapunov equations:

$$\left. \begin{array}{l} AX + XA^T + BB^T = 0 \\ XA + A^TX + C^TC = 0 \end{array} \right\} \text{In the continuous-time case} \quad (1.5.4)$$

and

$$\left. \begin{array}{l} AXA^T - X + BB^T = 0 \\ A^T X A - X + C^TC = 0 \end{array} \right\} \text{In the discrete-time case.} \quad (1.5.5)$$

It is numerically desirable that L is found without explicitly computing the matrix X and without forming the matrix C^TC or BB^T . Such an indirect method was found by Hammarling (1982). The Hammarling method both for the continuous-time and the discrete-time systems are described in **Chapter 8 (Algorithms 8.6.1 and 8.6.2)**.

The continuous-time algebraic Riccati equation (CARE):

$$XA + A^TX - XBR^{-1}B^TX + Q = 0 \quad (1.5.6)$$

and the discrete-time algebraic Riccati equation (DARE):

$$A^T X A - X - A^T X B (R + B^T X B)^{-1} B^T X A + Q = 0 \quad (1.5.7)$$

arise in

- LQR and LQG Design (**Chapter 10 and Chapter 12**)
- Kalman Filtering (**Chapter 12**)
- H_∞ Control (**Chapter 10**)

An algebraic Riccati equation may have many solutions. Of special interests, from applications viewpoints, is the unique stabilizing solution. **Numerical methods for computing such a solution are described in Chapter 13.** The stabilizing solution of the CARE may be obtained by constructing a basis of the invariant subspace corresponding to the eigenvalues with negative real parts (stable invariant subspace) of the associated Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}, \text{ where } S = BR^{-1}B^T.$$

It is natural to construct such a basis by finding the eigendecomposition of H . However, the eigenvector matrix can be highly ill-conditioned if H has multiple or near multiple eigenvalues. The difficulty can be overcome by using an ordered real Schur decomposition of H . This gives rise to the **Schur method for the CARE** (Laub (1979)).

Thus, if

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$$

is such that $U^T H U$ is an **ordered real Schur decomposition** of H :

$$U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix},$$

where the eigenvalues of H with negative real parts are contained in T_{11} , then $X = U_{21} U_{11}^{-1}$ is the stabilizing solution of the CARE.

The Schur method is widely used and is *numerically stable for all practical purposes*. In case of the DARE, the symplectic matrix

$$M = \begin{pmatrix} A + S(A^{-1})^T Q & -S(A^{-1})^T \\ (-A^{-1})^T Q & (A^{-1})^T \end{pmatrix}$$

takes the role of the Hamiltonian matrix. Since A has to be invertible, neither the eigenvector method nor the Schur method works for the DARE if A is singular. Even in case A is theoretically nonsingular but is computationally close to a singular matrix, these methods can be problematic for the DARE.

The Schur method, both for the CARE and the DARE, may not give an accurate solution in case R is nearly singular. This difficulty can be overcome by using an extended matrix pencil. For the CARE, the extended pencil is $P_{CARE}^E - \lambda N_{CARE}^E$, where

$$P_{CARE}^E = \begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix}$$

and

$$N_{CARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

It was further shown by Van Dooren (1981) that this $(2n+m) \times (2n+m)$ pencil can be compressed into an $2n \times 2n$ pencil by using the QR factorization of $\begin{pmatrix} R \\ B \end{pmatrix}$ without affecting the deflating subspace. The stabilizing solution of the CARE then is computed by finding the ordered generalized Schur decomposition of this compressed pencil using the QZ iteration algorithm (see **Algorithm 13.5.3**).

Analogous results exist for the DARE. For details, see **Chapter 13 (Algorithm 13.5.4)**. A method, based on an ordered real Schur decomposition of an extended pencil of the above type, is called an **inverse-free generalized Schur method**.

Another class of methods, called the **matrix sign-function methods**, has been developed. The matrix sign-function method for the CARE is based on computing the matrix sign-function of the Hamiltonian matrix H (see **Algorithm 13.5.6**). For the DARE, the matrix

$$H' = (P + N)^{-1}(P - N),$$

where

$$P = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix},$$

and

$$N = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}$$

is formed first and then the matrix sign-function method for the CARE is applied to H' (see **Algorithm 13.5.7**).

The matrix sign-function methods are not stable in general, unless an iterative refinement technique is used.

Any Riccati equation solver should be followed by an iterative refinement method, such as Newton's method. For detailed descriptions of **Newton's methods**, see Chapter 13 (**Section 13.5.4**). Newton's methods for the CARE and DARE are described, respectively, in **Algorithm 13.5.8** and **Algorithm 13.5.10**, while **Algorithm 13.5.9** and **Algorithm 13.5.11**, described **Newton's methods with line-search** for the CARE and the DARE.

In summary, *in case R is robustly nonsingular, the Schur method or the matrix sign function method, followed by Newton's method, is recommended for the CARE. In case R is nearly singular, the inverse-free generalized Schur method (**Algorithm 13.5.3**) should be used.*

For the DARE, *the inverse-free generalized Schur method (**Algorithm 13.5.4**) is the most general purpose method and is recommended to be used in practice. Again, the method should be followed by Newton's iterative refinement technique.*

1.6 Realization and Identification (Chapter 9)

Given a set of large number of Markov parameters, the problem of determining the system matrices A, B, C and D from this set, is called a state-space realization problem.

There are many realizations corresponding to a given set of Markov parameters and the one of the least possible dimension of A , called a **Minimal Realization** (MR), is of practical interest.

*A realization is a minimal realization if and only if it is both controllable and observable. (**Theorem 9.2.1**).*

The two minimal realizations are related via a nonsingular transforming matrix (Theorem 9.2.2**)** and the degree of a minimal realization is called the **McMillan degree**.

The existing algorithms for finding a minimal realization are all based on factoring an associated block Hankel matrix of Markov parameters:

$$M_k = \begin{pmatrix} H_1 & H_2 & \cdots & H_k \\ H_2 & H_3 & \cdots & H_{k+1} \\ \vdots & & & \\ H_k & H_{k+1} & \cdots & H_{2k-1} \end{pmatrix}$$

(k has to be greater than or equal to the McMillan degree), where $H_i = CA^{i-1}B$ is the i th Markov parameter.

The block Hankel matrix M_k can be highly ill-conditioned and therefore, care should be taken in obtaining its factorization. The singular value decomposition (SVD) is certainly a numerically viable procedure for such a factorization. *Two SVD based algorithms (Algorithms 9.3.1 and 9.3.2)* are presented in Chapter 9.

The Markov parameters are easily generated from a transfer function matrix in case they are of a discrete-time system; indeed in this case they are just the impulse responses. However, they are not readily available for a continuous-time system.

Thus, it is more of a practical importance to identify the system matrices directly from the input-output sequence.

A subspace identification algorithm (Algorithm 9.4.1) from DeMoor et al (1999), that does not require explicit computations of Markov parameters, is presented in Section 9.4.

Also stated in this chapter is a subspace algorithm (**Algorithm 9.4.2**) for *frequency-domain identification*. The frequency-domain identification problem concerns finding the system matrices A, B, C , and D from a given set of measured frequency responses at a set of frequencies (not necessarily distinct).

The subspace algorithms described here are projection-based algorithms and always consist of two steps. In the first step, a projection of certain subspaces generated directly from the input-output data is made to find an estimate of the states of the unknown system and then in the second step, the system matrices **are approximated** from the computed states explicitly or implicitly.

The subspace methods are numerically stable practical methods for systems identification.

1.7 Feedback Stabilization and Eigenvalue Assignment (Chapter 10 and Chapter 11)

Suppose that the uncontrolled system (1.3.1) is not stable, then it is desirable to make it stable. If the state vector $x(t)$ is measurable, then choosing

$$u(t) = -Kx(t),$$

we obtain the **closed-loop system**:

$$\dot{x}(t) = (A - BK)x(t).$$

Mathematically, the problem is then to find matrix K such that $A - BK$ is stable.

The system (1.0.1) is said to be **stabilizable** if such a K exists.

In many practical instances, just stabilizing a system is not enough. Certain design constraints require that all the eigenvalues be placed in certain specified regions of the complex plane.

This gives rise to the well-known **Eigenvalue Assignment** (EVA) problem or the so-called **pole-placement** problem:

Find a matrix K such that the matrix $A - BK$ has a desired set of eigenvalues.

A well-known result on the existence and uniqueness of the solution of the EVA problem is:

The EVA problem has a solution for any arbitrary set of desired eigenvalues if and only if (A, B) is controllable. In case the solution exists, it is unique if and only if the problem is a single-input problem.

Computing the feedback vector f via controller canonical form or using the related well-known Ackermann formula for single-input problem does not yield to a numerically viable algorithm. The reason is that the process of reduction to the controller canonical form is a numerically unstable process, because of the possible ill-conditioning of the transforming matrix.

Example 11.2.1 shows how *Ackermann's formula can give a set of closed-loop eigenvalues that are completely different from those were to be assigned*; however, when the same example was worked out using other EVA algorithms, such as the Kautsky-Nichols-Van Dooren algorithm (implemented in MATLAB function **place** or the simple recursive algorithm (**Algorithm 11.2.1**)), the computed closed-loop eigenvalues were quite accurate.

Several numerically viable algorithms, based on the reduction of the pair (A, B) to the *controller-Hessenberg form* rather than the *controller-Canonical form*, or to the *real Schur form* of A have been developed in recent years and a few selected algorithms are presented in **Chapter 11**. These include

- Recursive algorithms (**Algorithm 11.2.1** and **Algorithm 11.3.1**), based on evaluations of some simple recursive relations
- *QR* and *RQ* type algorithms (**Algorithm 11.2.2**, **Algorithm 11.2.3** and the one described in **Section 11.3.2**)
- The Schur algorithm (**Algorithm 11.3.3**) based on reduction of A to a real Schur form
- The robust eigenvalue assignment algorithm (**Algorithm 11.6.1**), which assigns not only a desired set of eigenvalues, but also uses the available freedom to make the eigenvectors as well-conditioned as possible.

Lyapunov and Sylvester equations can also be used for feedback stabilization and eigenvalue assignment. Two Lyapunov based methods for feedback stabilization; one for the continuous-time system and the other for the discrete-time system, have been described in Chapter 10 (**Section 10.2**). A solution of the EVA problem using a Sylvester equation, called the **Sylvester-observer equation**, has been developed by Bhattacharyya and deSouza (1982) (see **Exercise 12 of Chapter 11** for a description of this algorithm). A comparative study in tabular forms with respect to the efficiency, and numerical stability of different algorithms for EVA is given in Chapter 11 (**Sections 11.7 and 11.8**). *Based on factors, such as, ease of implementation, efficiency, and practical aspect of numerical stability, the author's favorites are: Algorithm 11.2.1 for single-input problem and Algorithm 11.3.3 for multi-input problem.* Note that **Algorithm 11.3.1** and the explicit QR algorithm described in **Section 11.3.2** for multi-input problem are more efficient than **Algorithm 11.3.3**; but there is a possibility that both these algorithms might give complex feedback matrices. Furthermore, **Algorithm 11.3.3** can be used for partial eigenvalue assignment that concerns reassigning only “bad” eigenvalues leaving the “good” ones unchanged.

1.8 State Estimation (Chapter 12)

In many practical situations, the states are not fully accessible and all that the designer knows are the input $u(t)$ and the output $y(t)$. However, for stabilization and eigenvalue assignment by state feedback, for *LQR* and *LQG* design, for Kalman filters, to solve H_∞ state-feedback control problems, and others, the knowledge of the complete state vector $x(t)$ is required. Thus, the unavailable states, somehow, need to be estimated accurately from the knowledge of the matrices A , B , and C and the input and output vectors $u(t)$ and $y(t)$. Mathematically, the **state estimation problem is the problem of finding an estimate $\hat{x}(t)$ of $x(t)$ such that the error vector $e(t) = x(t) - \hat{x}(t)$ approaches zero as fast as possible.**

It is shown (**Theorem 12.2.1**) that if the states $x(t)$ of the system (1.0.1-1.0.2) are estimated by

$$\dot{\hat{x}}(t) = (A - KC)\hat{x}(t) + Ky(t) + Bu(t), \quad (1.8.1)$$

where the matrix K is constructed such that $A - KC$ is a stable matrix, then the error vector $e(t)$ has the property that $e(t) \rightarrow 0$ as $t \rightarrow \infty$. The observability of the pair (A, C) ensures the existence of such a matrix K .

It is clear from the above result that the state estimation problem can be solved by solving the feedback stabilization or the eigenvalue assignment problem for the pair (A^T, C^T) .

An alternative approach for state estimation is via solution of the Sylvester equation $XA - FX = GC$. It is shown in **Theorem 12.3.1** that if an observer $z(t)$ is constructed satisfying

$$\dot{z}(t) = Fz(t) + Gy(t) + Pu(t),$$

such that

$$(i) \quad XA - FX = GC \quad (1.8.2)$$

$$(ii) P = XB \quad (1.8.3)$$

$$(iii) F \text{ is stable}, \quad (1.8.4)$$

then the error vector $e(t) = z(t) - Xx(t) \rightarrow 0$ for any $x(0), z(0)$, and $u(t)$. The estimate $\hat{x}(t)$ is given by $\hat{x}(t) = X^{-1}z(t)$.

The implementation of the above idea clearly requires a nonsingular solution of the Sylvester equation (1.8.2).

The equation (1.8.2) is called the **Sylvester-observer equation**, because it is a variation of the classical Sylvester equation (1.5.3) and arises in design of an observer.

In case the matrix C has full rank r , the full-order state estimation problem can be reduced to a reduced-order problem of dimension $n - r$. In this case, only those linear combinations of the states which are not available in $y(t) = Cx(t)$ are reconstructed.

Again, both eigenvalue assignment and the Sylvester-observer matrix equation approaches can be used to construct a reduced-order observer.

Thus, if the Sylvester-observer equation approach is used, and if (1.8.2)-(1.8.4) are satisfied, then

$$\hat{x}(t) = \begin{pmatrix} X \\ C \end{pmatrix}^{-1} \begin{pmatrix} z(t) \\ y(t) \end{pmatrix},$$

provided the matrix $\begin{pmatrix} X \\ C \end{pmatrix}$ is nonsingular. Note that in this case, the matrices X, F and C , and G are of dimensions $(n - r) \times n, (n - r) \times (n - r), r \times n$ and $n \times r$, respectively.

Two numerically reliable algorithms (**Algorithm 12.7.1** and **Algorithm 12.7.2**) for the multi-output case, both based on the reduction of the pair (A, C) to *controller-Hessenberg forms*, have been described in Chapter 12. Furthermore, necessary and sufficient conditions for the non singularity of the solution X of the Sylvester-observer equation have been given in **Theorem 12.6.1** and **Theorem 12.6.2**.

Optimal State Estimation: The Kalman Filter

The problem of finding the optimal steady state estimation of the states of a stochastic system is considered in **Section 12.9**. An algorithm (**Algorithm 12.9.1**) for the state estimating using Kalman filter is described and the duality between Kalman filter and the LQR design is discussed in **Section 12.9**.

The Linear Quadratic Gaussian Problem

The linear quadratic Gaussian problem (LQG) deals with optimization of a performance measure for a stochastic system. An algorithm (**Algorithm 12.10.1**) for LQG design is described in **Section 12.10.1**.

1.9 Internal Balancing and Model Reduction (Chapter 14)

The model reduction is a procedure for obtaining a reduced-order model that preserves some important properties such as stability, and is close to the original model, in some sense. One way to obtain such a model is via **internally balanced realization**. A continuous-time stable system given by (A, B, C) is internally balanced if there exists a nonsingular transforming matrix T such that the controllability Grammian C_G and the observability Grammian O_G , respectively, given by

$$\begin{aligned} AC_G + C_G A^T + BB^T &= 0 \\ O_G A + A^T O_G + C^T C &= 0 \end{aligned}$$

are such that

$$T^{-1}C_G T^T = T^T O_G T = \Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_d, \sigma_{d+1}, \dots, \sigma_n).$$

The diagonal entries $\sigma_1, \dots, \sigma_n$ are called the **Hankel singular values**. Once the system is internally balanced, the reduced-order model can be obtained by deleting the states corresponding to the negligible Hankel singular values. It can be shown (**Theorem 14.4.1**) that if $G(s)$ and $G_R(s)$ denote the transfer function matrices, respectively, of the original and the reduced-order model, then the error $E = \|G(s) - G_R(s)\|_\infty$ is bounded and $G_R(s)$ is also stable.

An algorithm (**Algorithm 14.2.1**) due to Laub (1980) for constructing a balanced realization, based on the singular value decomposition of the matrix $L_o^T L_C$, where L_o and L_C are, respectively, the Cholesky factors of controllability and observability Grammians, is given in Section 14.2. *The difficulty with this method is that the transforming matrix T may be ill-conditioned.* The following example due to Safanov and Chiang (1989) illustrates this point.

$$\text{Let } \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) = \left(\begin{array}{cc|c} -\frac{1}{2} & -\epsilon & \epsilon \\ 0 & -1\frac{1}{2} & 1 \\ \hline 1 & \epsilon & 0 \end{array} \right).$$

Then the transforming matrix T of the internal balancing algorithm is given by

$$T = \left(\begin{array}{cc} \sqrt{\frac{1}{\epsilon}} & 0 \\ 0 & \sqrt{\epsilon} \end{array} \right).$$

It is clear that T becomes progressively ill-conditioned as ϵ becomes smaller and smaller. An algorithm (**Algorithm 14.4.2**), based on the Schur decomposition of the matrix $C_G O_G$, that overcomes this difficulty is described in **Section 14.4**. The Schur algorithm was developed by Safanov and Chiang (1989). It produces a reduced-order model which has the same error property as the one obtained via internal balancing.

This chapter also contains several other algorithms for balanced realization and model reduction, including the **Square-root algorithm** (**Algorithm 14.2.2**) for balanced realization and **Hankel-norm approximation algorithm** for model reduction (**Algorithm 14.5.1**).

1.10 Krylov Subspace Methods for Large and Sparse Control Problems (Chapter 15)

Numerical methods described in Chapters 5-14 are not suitable for sparse and structured problems. This is because, these methods are based on reduction of the system matrices to canonical forms which are obtained by using Householder's and Givens' transformations, and QR and QZ iteration techniques; and such techniques are well-known to destroy the sparsity structures. On the other hand, there are practical control problems arising from discretizations of the distributed parameter models (e.g., *large space structures, power systems and others*) which are very large and typically sparse. The computational complexity and storage requirements can grow exponentially with the dimensions of these problems.

To deal with these large and sparse problems, specialized methods that can take advantage of the sparsity, are required.

A few such methods have been developed in the last few years by taking advantage of the recent advances in large and sparse matrix computational techniques in numerical linear algebra. Some of these methods are described in Chapter 15. These include the *Arnoldi-based methods* for the *Lyapunov equation* (**Algorithms 15.5.1-15.5.3**), for the *Sylvester and Sylvester-observer equations* (**Algorithms 15.5.4 - 15.5.6**), for the *continuous-time Algebraic Riccati equation* (**Algorithm 15.5.7**), for the *partial eigenvalue assignment* (**Algorithm 15.5.8**), and *Lanczos and Arnoldi methods for model reduction* (**Algorithm 15.5.9 - 15.5.11**). *This is an emerging area of research and further developments in this area are in progress.*

1.11 Numerical Methods for Control Problems Modeled by Systems of Second-Order Differential Equations (Chapter 16)

Vibrating structures such as buildings, bridges, highways, etc., are distributed parameter systems. Very often, due to the lack of effective numerical techniques to deal directly with distributed parameter systems, these systems are discretized using techniques of finite element or finite differences yielding a second-order model of the form:

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = 0, \quad (1.11.1)$$

where

$$M = M^T > 0 \text{ (mass matrix)}$$

$$K = K^T \text{ (stiffness matrix)}$$

$$D = D^T \text{ (damping matrix)}$$

Dangerous oscillations, called **resonances**, will occur when one or more of the eigenvalues, called **natural frequencies**, of the associated quadratic matrix pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ becomes equal or nearly equal to an external frequency (see Datta (1995), Inman (1989)). One way to combat such dangerous oscillations is to apply a control force of the form $Bu(t)$ to the structure, giving rise to the second-order control system:

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = Bu(t). \quad (1.11.2)$$

Two standard approaches to solve a control problem for the second-order system are (i) to transform the second-order control system to a standard first-order state-space form and then apply a well-established first-order technique to the transformed system, (ii) to apply the well-known *independent modal space control* (IMSC) approach.

There are numerical and engineering difficulties with both these approaches. Reformulation of (1.11.2) to the *standard first-order state-space form*

$$\begin{pmatrix} \dot{q}(t) \\ \ddot{q}(t) \end{pmatrix} = \begin{pmatrix} O & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix} \begin{pmatrix} q(t) \\ \dot{q}(t) \end{pmatrix} + \begin{pmatrix} O \\ M^{-1}B \end{pmatrix} u(t),$$

requires an explicit inversion of the matrix M , and thus *the state matrix may not be accurately determined if M is ill-conditioned.* Furthermore, the exploitable structures such as symmetry, definiteness, sparsity, and others, very often offered by practical problems, are completely destroyed under such transformations.

A non-standard first-order representation of system (1.11.2) is of the form

$$\begin{pmatrix} -K & O \\ O & M \end{pmatrix} \dot{z}(t) = \begin{pmatrix} O & -K \\ -K & -D \end{pmatrix} z(t) + \begin{pmatrix} O \\ B \end{pmatrix} u(t),$$

which is known as a **descriptor control system** and *the effective numerical methods for many of the descriptor control problems are not yet fully developed.*

A basic idea behind the IMSC approach is to decouple the problem into n easy-to-solve independent problems, and then piece the solutions of the individual problems together to obtain the solution of the original problem. The decoupling of the left hand side of (1.11.2) is easily achieved if the matrices M , K , and D are simultaneously diagonalized. Unfortunately, however, a very stringent unpractical commutativity relation $DM^{-1}K = KM^{-1}D$ needs to be satisfied for simultaneous diagonalization of these three matrices. An ad-hoc damping, called the **Rayleigh damping** or the **modal damping** $D = \alpha M + \beta K$, is very often used for computational convenience. The Rayleigh damping is also unpractical (William and Laub (1992)). Moreover, even if the left hand side of (1.11.2) is decoupled, it does not necessarily decouple the right hand side control input $S_m^T Bu$, where S_m is the transforming matrix that transforms the matrices M , K , D simultaneously to diagonal forms. For that, another sets of stringent requirements on the location and number of sensors and actuators are required (see Inman (1989), p. 173 for details).

To deal with these difficulties, a “**direct and nonmodal**” technique for **feedback stabilization** and a “**direct and partial modal**” technique for **partial eigenvalue and eigenstructure assignment problems** have been recently developed (Datta, Elhay, and Ram (1997), Datta and Sarkissian (1999, 2001), Datta, Elhay, Ram, and Sarkissian (2000)). The *feedback stabilization (partial eigenvalue assignment)* problem for the second-order control system (1.10.2) is defined as follows:

Given the matrices M , K , and D , and the control matrix B , find feedback matrices F and G such that the closed-loop quadratic pencil $P_C(\lambda) = \lambda^2 + \lambda(D - BF^T) + (K - BG^T)$ is asymptotically stable (has a set of $p << 2n$ desired eigenvalues and the remaining $2n - p$ eigenvalues are the same as those of the open-loop pencil $P(\lambda) = \lambda^2M + \lambda D + K$).

The *partial eigenstructure assignment* concerns assignment of not only a given set of eigenvalues but a set of the associated eigenvectors as well, leaving the remaining eigenvalues and eigenvectors unchanged.

The techniques are **direct**, because the problems are solved directly in the second-order setting without resorting to a first-order reformulation.

The feedback stabilization technique is **nonmodal** in the sense that no knowledge of eigenvalues and eigenvectors of the pencil $P(\lambda)$ is required to implement this technique. Similarly, the **partial modal technique** for partial eigenvalue and eigenstructure assignment requires only those small number of eigenvalues and eigenvectors that are required to be reassigned; furthermore, it has been established mathematically that the remaining $2n - p$ eigenvalues and the corresponding eigenvectors remain invariant. **These techniques are described in Chapter 16 (Algorithm 16.7.1-16.7.6).**

Specifically, **Algorithm 16.7.1** is a non-modal stabilizing algorithm; **Algorithms 16.7.2 and 16.7.3** are partial-modal algorithms for single-input and multi-input partial eigenvalue assignment problems, respectively; **Algorithm 16.7.4** is a partial-modal algorithm for partial eigenstructure assignment problem; and **Algorithms 16.7.5 and 16.7.6** are **robust eigenvalue assignment** algorithms. **Because of the direct and partial-modal nature, these algorithms are very suitable for practical applications.** This chapter also contains results on **stability, controllability** (Section 16.5) and **eigenvalue bounds, orthogonality of the eigenvectors** of the quadratic pencil (Section 16.6).

1.12 Nearness to Uncontrollability and Instability (Chapter 6 and Chapter 7) and Robust Stability and Stability Radius (Chapter 7 and Chapter 10)

1.12.1 Nearness to Uncontrollability and Instability

There are systems, which are, theoretically perfectly controllable, but may be very close to uncontrollable systems. A well-known example is the system with A as an upper Hessenberg matrix having 1's along the subdiagonal and -1 everywhere else, and $B = (1, 0, \dots, 0)^T$. The system is clearly controllable, but becomes close to an uncontrollable system, when n is large. *Thus, what is important in practice is to know when a system is close to an uncontrollable system rather than asking if it is controllable or not.*

A measure of distance to uncontrollability, denoted by $\mu(A, B)$, is defined (Paige (1980)) as

follows:

$$\mu(A, B) = \min\{\|\Delta A, \Delta B\|_2 \text{ such that the system } (A + \Delta A, B + \Delta B) \text{ is uncontrollable}\}$$

It can be shown (**Theorem 6.9.1**) (Miminis (1981), Eising (1984), Kenney (1998)) that

$$\mu(A, B) = \min \sigma_n(sI - A, B),$$

where σ_n denotes the smallest singular value of M . Several algorithms (Elsner and He (1991), Miminiis (1981), Eising (1982), Bolley (1987), Wicks and DeCarlo (1990)) for computing $\mu(A, B)$ have been developed in the last several years. A Newton-type algorithm (**Algorithm 6.9.1**) due to Elsner and He (1991) and an SVD algorithm due to Wicks and DeCarlo (**Algorithm 6.9.2**) are described in Chapter 6.

Similar remarks hold for the stability of a system. *There are systems which are clearly stable theoretically, but in reality are very close to unstable systems.* A well-known example of such a system is the system with a 20×20 upper bidiagonal matrix A having 10's along the subdiagonal and -1 along the main diagonal. Since the eigenvalues of A are all -1, it is perfectly stable. However, if the (20, 1)th entry is perturbed to $\epsilon = 10^{-18}$ from zero, then one of the eigenvalues becomes positive, making the matrix A unstable.

A measure of the distance to instability is

$$\beta(A) = \min\{\|\Delta A\| \text{ such that } A + \Delta A \text{ is unstable}\}.$$

Again, it can be shown (Van Loan (1985)) that

$$\beta(A) = \min_{\omega \in \mathbb{R}} \sigma_{\min}(A - j\omega I).$$

A bisection algorithm (**Algorithm 7.6.1**) due to Byers (1988) for estimating $\beta(A)$ is described in **Chapter 7**.

The algorithm is based on the following interesting result (**Theorem 7.6.1**):

$$\text{Define } H(\sigma) = \begin{pmatrix} A & -\sigma I \\ \sigma I & -A^* \end{pmatrix}.$$

Then $\beta(A) \leq \sigma$ if and only if $H(\sigma)$ has a purely imaginary eigenvalue.

A bisection algorithm (**algorithm 7.6.2**) for estimating the distance to a discrete unstable system is also described in this Chapter.

1.12.2 Robust stability and stability Radius (Chapter 7 and Chapter 10)

The robust stability concerns the stability of the perturbed system:

$$\dot{x}(t) = (A + E)x(t),$$

where A is a stable matrix and E is an $n \times n$ perturbation matrix. Several robust stability results (**Theorems 7.7.1-7.7.4**) and a robust stability algorithm (**Algorithm 7.7.1**) using Lyapunov equations are given in Chapter 7.

The stability radius of the matrix triple (A, B, C) is defined as

$$r_{\mathbb{F}}(A, B, C) = \inf\{\bar{\sigma}(\Delta) : \Delta \in \mathbb{F}^{m \times r} \text{ and } A + B\Delta C \text{ is unstable}\}, \quad (7.8.1)$$

where $\bar{\sigma}(M)$, following the notation of Qiu et al (1995), denotes the largest singular value of M (that is, $\bar{\sigma}(M) = \sigma_{max}(M)$). For real matrices (A, B, C) , $r_{\mathbb{R}}(A, B, C)$ is called the **real stability radius** and, for complex matrices (A, B, C) , $r_{\mathbb{C}}(A, B, C)$ is called the **complex stability radius**.

The stability radius, thus, determines the magnitude of the smallest perturbation needed to destroy the stability of the system.

“Stability” here is referred to as either continuous-stability (with respect to the left half-plane) or discrete-stability (with respect to the unit circle).

Let $\partial\mathbb{C}_g$ denote the boundary of either the half plane or the unit circle. Let A be stable or discrete-stable.

Formulas for complex and real stability radiiuses are given, respectively, in **Theorem 7.8.1** and **Theorem 7.8.2**.

Section 10.7 of Chapter 10 deals with the relationship between the complex stability radius and Riccati equation. A **characterization of the complex stability radius** is given in **Theorem 10.7.2** using a parametric Hamiltonian matrix and the connection between complex stability radius and an algebraic Riccati equation is established in **Theorem 10.7.3**.

A simple bisection algorithm (**Algorithm 10.7.1**) for computing the complex stability radius, based on Theorem 10.7.2, is then described at the conclusion of this section.

1.13 Sensitivity and Condition Numbers of Control Problems

The sensitivity of a computational problem is determined by its condition number. If the condition number is too large, then the solution is too sensitive to small perturbations and the problem is called an **ill-conditioned** problem.

The ill-conditioning has a direct effect on the accuracy of the solution. *If a problem is ill-conditioned, then even with a numerically stable algorithm, the accuracy of the solution cannot be guaranteed.* Thus, it is important to know if a computational problem is ill- or well-conditioned.

While the condition numbers for major problems in numerical linear algebra have been identified (**Chapter 3**), only a few studies on the sensitivities of computational problems in control have been made so far. The sensitivity study is done by theoretical perturbation analysis.

In this book, we have included perturbation analysis of the **matrix exponential problem**, (**Section 5.3.2**), of the **Lyapunov and Sylvester equations** (**Section 8.3**), of the **algebraic**

Riccati equations (Section 13.4) and of the state feedback and eigenvalue assignment problems (Sections 11.4 and 11.5).

1.13.1 Conditioning of the Matrix Exponential Problem (Chapter 5)

A perturbation analysis for the matrix exponential problem, that is, the problem of computing e^{At} , has been done and the condition number for the problem has been identified by Van Loan (1977). The condition number for the problem e^{At} is given by

$$\kappa(A, t) = \max_{\|E\|_2 \leq 1} \left\| \int_0^t e^{A(t-s)} E e^{As} ds \right\|_2 \frac{\|A\|_2}{\|e^{At}\|_2}.$$

It can be shown that $\kappa(A, t) \geq t \|A\|_2$ with equality holding for all nonnegative t if and only if A is normal. Thus $\|A\|_2$ gives an idea about the conditioning of the matrix e^A .

1.13.2 Conditioning of the Lyapunov and Sylvester Equations (Chapter 8)

Let $A \in \mathbb{R}^{m \times m}$ and $B \in \mathbb{R}^{n \times n}$. Then the Sylvester equation (1.5.3) is equivalent to the algebraic system (**Equation 8.2.1**):

$$P \text{vec}(x) = \text{vec}(C),$$

where $P = (I_m \otimes B + A^T \otimes I_n)$, and it has been shown (**Theorem 8.3.2**) by Higham (1996) that the number ψ defined below is the **condition number** of the Sylvester equation (1.5.3):

$$\psi = \|P^{-1}[\beta(X^T \otimes I_n, \alpha(I_m \otimes X), -\gamma I_m n)]\|_2 / \|X\|_F,$$

where α, β , and γ are certain tolerances on the perturbations of A, B , and C , respectively. Here and elsewhere $M \otimes N$ denotes the Kronecker product of the matrices M and N .

The Lyapunov equation is a special case of the Sylvester equation. For an analogous expression of the condition number of the Lyapunov equation, see **Section 8.3**.

A crude bound of the relative error in the solution of the Sylvester equation can be found in **Theorem 8.3.1** in terms of the perturbation bounds of A, B, C , and the quantity $\text{sep}(B, -A)$ defined by

$$\text{sep}(B, -A) = \min_{X \neq 0} \frac{\|XA + BX\|_F}{\|X\|_F} = \|P^{-1}\|_2 = \frac{1}{\delta_{\min}(P)}.$$

If A is stable, then the sensitivity of the Lyapunov equation can be determined (**Theorem 8.3.3**) by simply solving the Lyapunov equation $HA + A^T H = -I$.

The result is due to Hewer and Kenney (1988). Furthermore, it can be shown (**Theorem 8.3.6**) that *the Sylvester equation is ill-conditioned if A and B are ill-conditioned. Thus, in particular, if A is nearly singular, then the Lyapunov equation is ill-conditioned* (Ghavimi and Laub (1995)).

Since computing the condition numbers of the Sylvester and Lyapunov equation are computationally intensive and expensive, it is useful to have a condition number estimators that can be more cheaply obtained.

An algorithm (**Algorithm 8.3.1**) due to Byers (1984) for estimating $\text{sep}(A, B^T)$ is described in **Chapter 8**. The major computational requirement of this algorithm is the solution of a single Sylvester equation.

Perturbation results on the discrete Lyapunov equation $A^T X A - X = C$ are contained in **Theorems 8.3.4 and 8.3.5**.

1.13.3 Conditioning of the Algebraic Riccati Equations (Chapter 13)

A result due to Sun (1998) on the relative error in the solution X of the CARE (**Equation 1.5.6**) in terms of the relative perturbations in A, Q , and $S = BR^{-1}B^T$, is stated in **Theorem 13.4.1**.

Using this result, three relative condition numbers with respect to Q, A , and S , denoted, respectively, by $\kappa_{\text{CARE}}^{\text{REL}}(Q), \kappa_{\text{CARE}}^{\text{REL}}(A)$ and $\kappa_{\text{CARE}}^{\text{Rel}}(S)$, are defined. They are (*in unitarily invariant norm*):

$$\kappa_{\text{CARE}}^{\text{REL}}(Q) = \frac{\|Q\|}{l\|X\|}, \quad \kappa_{\text{CARE}}^{\text{REL}}(A) = \frac{p\|A\|}{\|X\|}, \quad \text{and} \quad \kappa_{\text{CARE}}^{\text{REL}}(S) = \frac{q\|S\|}{\|X\|},$$

where

$$l = \|T^{-1}\|_2, \quad p = \|T^{-1}(I_n \otimes X + (X^T \otimes I_n)E)\|_2$$

and

$$q = \|T^{-1}(X^T \otimes X)\|_2$$

with

$$T = I_n \otimes (A - SX)^T + (A - SX)^T \otimes I_N$$

and

$$E = \sum_{i,j=1}^n (e_i e_j^T) \otimes (e_j e_i^T).$$

In Frobenius norm $l = \|\Omega^{-1}\|^{-1}$, where $\|\Omega^{-1}\|_F = \frac{1}{\text{sep}((A - SX)^T - (A - SX))}$.

An approximate condition number due to Byers (1985) is:

$$\kappa_{\text{CARE}}^B = \frac{1}{\|X\|_F} \left(\frac{\|Q\|_F}{l} + p\|A\|_F + q\|S\|_F \right),$$

and upper and lower bounds for this approximate condition number have been obtained by Kenney and Hewer (1990) using the solutions of the Lyapunov equations:

$$(A - SX)^T H_k + H_k (A - SX) = -X^k, \quad k = 0, 1, 2.$$

From these bounds, *it can be shown, in general, the CARE will be ill-conditioned if the matrices H_0, H_1 , and H_2 have large norms*. For details, see **Section 13.4.1**.

The conditioning of the DARE has been discussed in **Section 13.4.2** (see (13.4.16)). Using first-order perturbations only, an **approximate condition number** for the DARE has been

identified and is given by

$$2 \| A \|_F^2 \frac{\| Q \|_F}{\| X \|_F} + \frac{\| A \|_F^2 \| S \|_F \| X \|_F}{sep_d(A_d^T, A_d)},$$

where $sep_d(A_d^T, A_d)$ is the minimum singular value of the matrix $A_d^T \otimes A_d - I_{n^2}$, with $A_d = A - B(R + B^T X B)^{-1} B^T X A$.

1.13.4 Conditioning of the Feedback and Eigenvalue Assignment Problems (Chapter 11)

The conditioning of the feedback problem and the eigenvalue assignment problem are two completely different matters. *Even though a feedback matrix has been computed using a numerically stable algorithm, there is no guarantee that the eigenvalues of the closed-loop matrix are close to those which are to be assigned.* See **Example 11.5.1**.

Perturbation results, both in single-input and multi-input cases, and the condition numbers of the feedback problem have been stated in **Section 11.4** of Chapter 11 (**Theorems 11.4.1-11.4.4**).

The conditioning of the closed-loop eigenvalues depends upon several interrelated factors, either individually or in combination. These include (i) *the conditioning of the feedback problem* (ii) *the condition number of the eigenvector matrix of the closed-loop system* (iii) *the distance to uncontrollability, and the distance between the closed-loop and open-loop eigenvalues, and* (iv) *the norm of the feedback matrix.*

In general, *the problem of assigning eigenvalues becomes progressively ill-conditioned with the increasing order of n* (He, Laub and Mehrmann (1995)).

However, the recent studies by Mehrmann and Xu (1988) and by Calvetti, Lewis and Reichel (1999) show that the *conditioning of the problem can be improved by choosing the poles judiciously using certain criteria in a compact set in the complex plane, rather than placing them arbitrarily at fixed points.*

1.14 H_∞ -Control (Chapter 10)

H_∞ -control problems concern stabilizing perturbed versions of the original system with certain constraints on the size of the perturbations. Both state feedback and output feedback versions of H_∞ -control have been considered in the literature and stated in **Chapter 10** of this book. We state below a simplified version of the output feedback H_∞ -control problem and give a result on the existence of a solution.

The H_∞ -Control Problem: Output Feedback Case

Given the perturbed system

$$\begin{aligned}\dot{x}(t) &= Ax(t) + B_1 w(t) + B_2 u(t) \\ z(t) &= C_1 x(t) + D_{12} u(t) \\ y(t) &= C_2 x(t) + D_{21} w(t)\end{aligned}$$

the transfer function from the inputs $\begin{bmatrix} w \\ u \end{bmatrix}$ to the outputs $\begin{bmatrix} z \\ y \end{bmatrix}$ is:

$$G(s) = \begin{pmatrix} 0 & D_{12} \\ D_{21} & 0 \end{pmatrix} + \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} (sI - A)^{-1} (B_1, B_2) = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}$$

If a feedback controller $K(s)$ is defined by $u = K(s)y$, then the closed-loop transfer function matrix $T_{zw}(s)$ from the disturbance w to the output z is

$$T_{zw} = G_{11} + G_{12}K(I - G_{22}K)^{-1}G_{21}.$$

The goal of the output feedback H_∞ control problem in this case is to find a controller $K(s)$ such that $\|T_{zw}(s)\|_\infty < \gamma$, where γ is a given positive real number.

Under the assumptions (i) (A, B_1) is stabilizable and (A, C_1) is detectable, (ii) (A, B_2) is stabilizable and (A, C_2) is detectable, (iii) $D_{12}^T(C_1, D_{12}) = [0, I]$, (iv) $\begin{pmatrix} B_1 \\ D_{21} \end{pmatrix} D_{21}^T = \begin{pmatrix} 0 \\ I \end{pmatrix}$; it can be shown (see Doyle, Glover, Khargonekar and Francis (1999)) that the H_∞ -control problem as stated above has a solution if there exist unique symmetric positive semidefinite stabilizing solutions X and Y , respectively, to the pair of algebraic Riccati equations

$$\begin{aligned} XA + A^TX - X(B_2B_2^T - \frac{1}{\gamma^2}B_1B_1^T)X + C_1^TC_1 &= 0 \\ AY + YA^T - Y(C_2^TC_2 - \frac{1}{\gamma^2}C_1^TC_1)Y + B_1B_1^T &= 0 \end{aligned}$$

and $\rho(XY) < \gamma^2$, where $\rho(XY)$ is the spectral radius of the matrix XY . Furthermore, in this case, one such controller is given by the transfer function

$$K(s) = -F(sI - \hat{A})^{-1}ZL,$$

where

$$\hat{A} = A + \frac{1}{\gamma^2}B_1B_1^TX + B_2F + ZLC_2$$

and

$$F = -B_2^TX, L = -YC_2^T, Z = (I - \frac{1}{\gamma^2}YX)^{-1}$$

Two numerical algorithms for computing H_∞ -norm of a stable transfer function matrix: the *bisection algorithm* (**Algorithm 10.6.1**) due to Boyd et al. (1989), and the *two-step algorithm* (**Algorithm 10.6.2**) due to Bruinsma, et al. (1990) are described in **Chapter 10**. Both these algorithms are based on the following well-known result (**Theorem 10.6.1**):

Let $G(s)$ be the transfer function matrix of the system (1.0.1)-(1.0.2) and let $\gamma > 0$ be given, then $\|G\|_\infty < \gamma$ if and only if $\sigma_{\max}(D) < \gamma$ and the matrix M_γ defined by

$$M_\gamma = \begin{pmatrix} A + BR^{-1}D^TC & BR^{-1}B^T \\ -C^T(I + DR^{-1}D^T)C & -(A + BR^{-1}D^TC)^T \end{pmatrix},$$

where $R = \gamma^2 I - D^T D$, has no imaginary eigenvalues.

The implementation of the algorithms require a lower and an upper bound for the H_∞ -norm. These bounds can be computed using the **Enns-Glover formula**:

$$\begin{aligned}\gamma_{lb} &= \max\{\sigma_{\max}(D), \sigma H_1\} \\ \gamma_{ub} &= \sigma_{\max}(D) + 2 \sum_{i=1}^2 \sigma H_i,\end{aligned}$$

where σ_{H_i} is the *i*th **Hankel singular value**. The Hankel singular value are the square-roots of the eigenvalues of the matrix $C_G O_G$, where C_G and O_G are, respectively, the controllability and observability Grammian.

1.15 Software for Control Problems

There now exist several high-quality numerically reliable softwares for control systems design and analysis. These include

- MATLAB-based Control Systems Tool Box
- MATHEMATICA-based Control System Professional - Advanced Numerical Methods (**CSP-ANM**).
- Fortran-based SLICOT (A Subroutine Library in Systems and Control Theory).
- MATRIX_X.
- The System Identification Toolbox
- MATLAB-based Robust Control Toolbox
- μ -Analysis and Synthesis Toolbox

A MATLAB-based tool-kit, called **MATCONTROL**, is provided with this book.

A feature that distinguishes MATCONTROL and CSP-ANM from the other software is that both these software have implemented more than one (typically several) numerically viable algorithms for any given problem. This feature is specially attractive for **control education** in the class-rooms; because, students, researchers, and teachers will have opportunity to compare one algorithm over the others with respect to efficiency, accuracy, easiness for implementation, etc., without writing routines for each algorithm by themselves.

There also exist some specialized software developed by individuals for special problems. These include **polepack** developed by George Miminis (1991), **robpole** developed by Tits and Yang (1996), **Sylvplace** developed by Varga (2000) for pole placement; **ricpack** developed by Laub et al. for Riccati equations, **HTOOLS** for H_∞ and H_2 synthesis problems developed by Varga and Ionescu ((1999)), etc.

For a brief description of each of these tool boxes, see Appendix A.

References. For references of the papers cited in this chapter, the readers are referred to **References** section of each individual chapter.

Part I

Review of Linear and Numerical Linear Algebra

- Chapter 2.** A Review of Some Basic Concepts and Results from Theoretical Linear Algebra
- Chapter 3.** Some Fundamental Tools and Concepts from Numerical Linear Algebra
- Chapter 4.** Canonical Forms Obtained via Orthogonal Transformations

Chapter 2

A REVIEW OF SOME BASIC CONCEPTS AND RESULTS FROM THEORETICAL LINEAR ALGEBRA

Contents

2.1	Introduction	35
2.2	Vectors	35
2.2.1	Orthogonality of Vectors and Subspaces \mathbb{R}	36
2.3	Matrices	36
2.3.1	Basic Concepts	36
2.3.2	Range and Nullspaces	39
2.3.3	Rank of a Matrix	39
2.3.4	The Inverse of a Matrix	40
2.3.5	The Generalized Inverse of a Matrix	41
2.3.6	Similar Matrices	41
2.3.7	Orthogonal Projection	41
2.4	Some Special Matrices	42
2.4.1	Diagonal and Triangular Matrices	42
2.4.2	Unitary (Orthogonal) Matrix	42
2.4.3	Permutation Matrix	43
2.4.4	Hessenberg (Almost Triangular) Matrix	43
2.4.5	Companion Matrix	44
2.4.6	Nondiagonal Matrix	44
2.4.7	The Jordan Canonical Form of a Matrix	45

2.4.8	Positive Definite Matrix	45
2.4.9	Block Matrices	46
2.5	Vector and Matrix Norms	47
2.5.1	Vector Norms	47
2.5.2	Matrix Norms	48
2.6	Norm Invariant Properties Under Unitary Matrix Multiplication .	50
2.7	Kronecker Product, Kronecker Sum and Vec Operation	51

Topics Covered

Basic Concepts for Theoretical Linear Algebra.

2.1 Introduction

Although a first course in linear algebra is a prerequisite for this book, for the sake of completeness, we establish some notations and quickly review the basic definitions and concepts on matrices and vectors in this chapter. Fundamental results on **vector and matrix norms** are described in some details. These results will be used frequently in the later chapters of the book. *The students can review material of this chapter, as needed.*

2.2 Vectors

An ordered set of numbers is called a **vector**; the numbers themselves are called the **components** of the vector. A vector v having n components has the form

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}.$$

For convenience, the vector v above will sometimes be written as $v = (v_1, v_2, \dots, v_n)^T$.

A vector in this form is referred to as a **column vector** and its transpose is a **row vector**. The set of all real n -vectors (that is, each vector having n components) is denoted by \mathbb{R}^n . Similarly, the set of all complex vectors is denoted by \mathbb{C}^n . The transpose of the vector v will be denoted by v^T . The complex conjugate transpose of the vector v will be denoted v^* ; that is, $v^* = (\bar{v})^T$. **Unless otherwise stated, a column vector will simply be called a vector.**

If u and v are two vectors with n components, then their **sum** $u + v$ is the vector $u + v = \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_n + v_n \end{pmatrix}$.

If α is a scalar, then αu is the vector $\alpha u = \begin{pmatrix} \alpha u_1 \\ \alpha u_2 \\ \vdots \\ \alpha u_n \end{pmatrix}$.

The **inner product** of two vectors u and v is the **scalar** given by

$$u^*v = \bar{u}_1v_1 + \bar{u}_2v_2 + \cdots + \bar{u}_nv_n.$$

The **length** of a vector v , denoted by $\|v\|$, is $\sqrt{v^*v}$; that is, the length of v (or the Euclidean length of v) is $\|v\| = \sqrt{|v_1|^2 + |v_2|^2 + \cdots + |v_n|^2}$.

A set of vectors $\{m_1, \dots, m_k\}$ is said to be **linearly dependent** if there exist scalars c_1, \dots, c_k , not all zero, such that $c_1m_1 + \dots + c_km_k = 0$ (zero vector, that is, a vector having all its components zero.) Otherwise, the set is called **linearly independent**. The set of all linear combinations of m_1, \dots, m_k will be denoted by $\text{span } \{m_1, \dots, m_k\}$.

2.2.1 Orthogonality of Vectors and Subspaces \mathbb{R}

The angle θ between two nonzero vectors u and v is given by

$$\cos(\theta) = \frac{u^*v}{\|u\|\|v\|}.$$

Two vectors u and v are **orthogonal** if $\theta = 90^\circ$, that is, if $u^*v = 0$. The symbol \perp is used to denote orthogonality. The set of vectors $\{x_1, x_2, \dots, x_k\}$ in \mathbb{C}^n are **mutually orthogonal** if $x_i^*x_j = 0$ for $i \neq j$, and **orthonormal** if $x_i^*x_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta function; that is $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i \neq j$.

Let S be a nonempty subset of \mathbb{C}^n . Then S is called a **subspace** of \mathbb{C}^n if $s_1, s_2 \in S$ implies $c_1s_1 + c_2s_2 \in S$, where c_1 and c_2 are arbitrary scalars. That is, S is a subspace if any linear combination of two vectors in S is also in S .

For every subspace there is a unique smallest positive integer r such that every vector in the subspace can be expressed as a linear combination of at most r vectors in the subspace; r is called the **dimension** of the subspace and is denoted by $\dim[S]$.

Any set of r linearly independent vectors from S of $\dim[S] = r$ forms a **basis** of the subspace. The **orthogonal complement** of a subspace S is defined by $S^\perp = \{y \in \mathbb{C}^n \mid y^*x = 0 \text{ for all } x \in S\}$.

The set of vectors $\{v_1, v_2, \dots, v_n\}$ form an **orthonormal basis** of a subspace S if these vectors form a basis of S and are orthonormal.

Two subspaces S_1 and S_2 of \mathbb{C}^n are said to be **orthogonal** if $s_1^*s_2 = 0$ for every $s_1 \in S_1$ and every $s_2 \in S_2$. Two orthogonal subspaces S_1 and S_2 will be denoted by $S_1 \perp S_2$.

2.3 Matrices

2.3.1 Basic Concepts

A collection of mn elements arranged in a rectangular array of m rows and n columns is called a **matrix** of order $m \times n$. It has the form

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}.$$

It is denoted by $A = (a_{ij})_{m \times n}$, or simply by $A = (a_{ij})$, where it is understood that $i = 1, \dots, m$ and $j = 1, \dots, n$. The set of all $m \times n$ matrices with real elements is denoted by $\mathbb{R}^{m \times n}$. The set of all $m \times n$ matrices with complex elements is denoted by $\mathbb{C}^{m \times n}$.

A matrix A having the same number of rows and columns is called a **square matrix**. The square matrix having 1's along the main diagonal and zeros everywhere else is called the **identity matrix** and is denoted by I . Sometimes a $p \times p$ identity matrix is denoted by I_p or by $I_{p \times p}$ in this text.

The sum of two matrices $A = (a_{ij})$ and $B = (b_{ij})$ is a matrix of the same order as A and B and is given by

$$A + B = (a_{ij} + b_{ij}).$$

If c is a scalar, then cA is a matrix given by

$$cA = (ca_{ij}).$$

The **transpose** of an $m \times n$ matrix is the $n \times m$ matrix $A^T = \begin{pmatrix} a_{11} & a_{21} & \cdots & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & \cdots & a_{m2} \\ \vdots & \vdots & & & \vdots \\ a_{1n} & a_{2n} & \cdots & \cdots & a_{mn} \end{pmatrix}$.

The **conjugate transpose** of A is the matrix $A^* = (\bar{A})^T$, where \bar{A} is the matrix obtained from A by taking the complex conjugates of its elements. The conjugate transpose of a matrix A is also very often denoted by the symbol A^H .

A square matrix A is **symmetric** if $A^T = A$. A square complex matrix A is **Hermitian** if $A^* = A$. The matrix A is **skew symmetric** if $A^T = -A$ and **skew Hermitian** if $A^* = -A$.

Let A be $m \times n$ and B be $n \times p$. Then their product AB is an $m \times p$ matrix given by

$$AB = \left(\sum_{k=1}^n a_{ik} b_{kj} \right); \quad i = 1, \dots, m; \quad j = 1, \dots, p.$$

Note that if b is a column vector, then Ab is a column vector. The **outer product** of the two vectors a and b is the matrix:

$$ab^* = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} (\bar{b}_1 \quad \bar{b}_2 \quad \cdots \quad \bar{b}_m) = \begin{pmatrix} a_1 \bar{b}_1 & \cdots & a_1 \bar{b}_m \\ a_2 \bar{b}_1 & \cdots & a_2 \bar{b}_m \\ \vdots & & \vdots \\ a_n \bar{b}_1 & \cdots & a_n \bar{b}_m \end{pmatrix}.$$

The Determinant of a Matrix

For every square matrix A , there is a unique number associated with the matrix called the **determinant** of A , which is denoted by $\det(A)$. For a 2×2 matrix A , $\det(A) = a_{11}a_{22} - a_{12}a_{21}$; for a 3×3 matrix $A = (a_{ij})$, $\det(A) = a_{11} \cdot \det(A_{11}) - a_{12} \cdot \det(A_{12}) + a_{13} \cdot \det(A_{13})$, where A_{1i} is a 2×2 submatrix obtained from A by eliminating the first row and the i^{th} column, $i = 1, 2, 3$. This can be easily generalized. For an $n \times n$ matrix $A = (a_{ij})$ we have

$$\begin{aligned} \det(A) &= (-1)^{i+1} a_{i1} \det(A_{i1}) + (-1)^{i+2} a_{i2} \det(A_{i2}) \\ &\quad + \cdots + (-1)^{i+n} a_{in} \det(A_{in}), \end{aligned}$$

where A_{ij} is the submatrix of A of order $(n - 1)$ obtained by eliminating the i^{th} row and j^{th} column.

Theorem 2.3.1 The following simple properties of $\det(A)$ hold:

- $\det(A^T) = \det(A)$
- If A is of order n , then $\det(\alpha A) = \alpha^n \det(A)$, where α is a scalar.
- $\det(AB) = \det(A) \cdot \det(B)$, where A and B are square matrices of the same order.
- If any two rows or two columns of A are identical, then $\det(A) = 0$.
- If B is a matrix obtained from A by interchanging two rows or two columns, then $\det(B) = -\det(A)$.

The Characteristic Polynomial, the Eigenvalues and the Eigenvectors of a Matrix
Let A be an $n \times n$ matrix. Then the polynomial $p_A(\lambda) = \det(\lambda I - A)$ is called the **characteristic polynomial**. The zeros of the characteristic polynomial are called the **eigenvalues** of A . This is equivalent to the following: $\lambda \in \mathbb{C}$ is an eigenvalue of A if and only if there exists a nonzero vector x such that $Ax = \lambda x$.

The vector x is called a **right eigenvector** (or just an **eigenvector**) of A . A nonzero vector y is called a **left eigenvector** if $y^* A = \lambda y^*$ for some $\lambda \in \mathbb{C}$.

If an eigenvalue of A is repeated s times, then it is called a multiple eigenvalue of multiplicity s . If $s = 1$, then the eigenvalue is a **simple eigenvalue**.

Definition 2.3.1 If $\lambda_1, \lambda_2, \dots, \lambda_n$ are the n eigenvalues of A , then $\max |\lambda_i|$, $i = 1, \dots, n$ is called the **spectral radius** of A . It is denoted by $\rho(A)$.

Invariant Subspaces

A subspace S of \mathbb{C}^n is called invariant subspace or **A -invariant** if $Ax \in S$ for every $x \in S$.

Clearly, an eigenvector x of A defines a one dimensional invariant subspace.

An A -invariant subspace $S \subseteq \mathbb{C}^n$ is called a **stable invariant subspace** if the eigenvectors in S correspond to the eigenvalues of A with negative real parts.

The Cayley-Hamilton Theorem

The **Cayley-Hamilton Theorem** states that the characteristic polynomial of A is an annihilating polynomial of A . That is, if $p_A(\lambda) = \lambda^n + a_1\lambda^{n-1} + \dots + a_nI$, then $p_A(A) = A^n + a_1A^{n-1} + \dots + a_nI = 0$.

Definition 2.3.2 An $n \times n$ matrix A having fewer than n linearly independent eigenvectors is called a **defective matrix**.

Example 2.3.1

The matrix $A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ is defective. It has only one eigenvector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

2.3.2 Range and Nullspaces

For every $m \times n$ matrix A , there are two important associated subspaces: the **range** of A , denoted by $R(A)$, and the **null space** of A , denoted by $N(A)$, defined as follows:

$$\begin{aligned} R(A) &= \{b \mid b = Ax \text{ for some } x\} \\ N(A) &= \{x \mid Ax = 0\}. \end{aligned}$$

The dimension of $N(A)$ is called the **nullity** of A and is denoted by $\text{null}(A)$.

2.3.3 Rank of a Matrix

Let A be an $m \times n$ matrix. Then the subspace spanned by the row vectors of A is called the **row space** of A . The subspace spanned by the columns of A is called the **column space** of A . The range of A , $R(A)$, is the same as the column space of A .

The **rank** of a matrix A is the dimension of the column space of A . It is denoted by $\text{rank}(A)$.

An $m \times n$ matrix is said to have **full column rank** if its columns are linearly independent. The **full row rank** is similarly defined. A matrix A is said to have **full rank** if it has either full row rank or full column rank. If A does not have full rank, it is called **rank deficient**.

Example 2.3.2

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

has full rank; $\text{rank}(A) = 2$

Example 2.3.3

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 4 \\ 0 & 0 \end{pmatrix}$$

is rank deficient; $\text{rank}(A) = 1$; $\text{null}(A) = 1$.

Some Rank Properties

Let A be an $m \times n$ matrix. Then

- $\text{rank}(A) = \text{rank}(A^T)$.
- $\text{rank}(A) + \text{null}(A) = n$.
- $\text{rank}(AB) \geq \text{rank}(A) + \text{rank}(B) - n$, where B is $n \times p$.
- $\text{rank}(BA) = \text{rank}(A) = \text{rank}(AC)$, where B and C are nonsingular matrices of appropriate order.
- $\text{rank}(AB) \leq \min\{\text{rank}(A), \text{rank}(B)\}$.
- $\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$.

2.3.4 The Inverse of a Matrix

An $n \times n$ matrix A is said to be invertible if there exists an $n \times n$ matrix B such that $AB = BA = I$. The inverse of A is denoted by A^{-1} . An invertible matrix A is often called **nonsingular**. An interesting property of the inverse of the product of two invertible matrices is: $(AB)^{-1} = B^{-1}A^{-1}$.

Theorem 2.3.2 For an $n \times n$ matrix A , the following are equivalent:

- (i) A is nonsingular.
- (ii) $\det(A)$ is nonzero.
- (iii) $\text{rank}(A) = n$.
- (iv) $N(A) = \{0\}$.
- (v) A^{-1} exists.
- (vi) A has linearly independent rows and columns.
- (vii) The eigenvalues of A are nonzero.
- (viii) For all x , $Ax = 0$ implies that $x = 0$.
- (ix) The system $Ax = b$ has a unique solution.

2.3.5 The Generalized Inverse of a Matrix

The (Moore-Penrose) **generalized inverse** of a matrix A , denoted by A^\dagger , is a unique matrix satisfying the following properties:

- $AA^\dagger A = A$
- $A^\dagger AA^\dagger = A^\dagger$
- $(AA^\dagger)^* = AA^\dagger$
- $(A^\dagger A)^* = A^\dagger A$.

Notes: If A is square and invertible, then $A^\dagger = A^{-1}$.

2.3.6 Similar Matrices

Two matrices A and B are called **similar** if there exists a nonsingular matrix T such that

$$T^{-1}AT = B.$$

An important property of similar matrices: **Two similar matrices have the same eigenvalues.** However, two matrices having the same eigenvalues need not be similar.

2.3.7 Orthogonal Projection

Let S be a subspace of \mathbb{C}^n . Then an $n \times n$ matrix P having the properties:

(i) $R(P) = S$, (ii) $P^* = P$ (P is **Hermitian**), (iii) $P^2 = P$ (P is **idempotent**)

is called the **orthogonal projection** onto S or simply the **projection matrix**. We denote the orthogonal projection P onto S by P_S . **The orthogonal projection onto a subspace is unique.**

Let $V = (v_1, \dots, v_k)$, where $\{v_1, \dots, v_k\}$ is an orthonormal basis for a subspace S . Then

$$P_S = VV^*$$

is the unique orthogonal projection onto S . **Note that V is not unique, but P_S is.**

A Relationship between P_S and P_{S^\perp}

If P_S is the orthogonal projection onto S , then $I - P_S$, where I is the identity matrix of the same order as P_S , is the orthogonal projection onto S^\perp . It is denoted by P_{S^\perp} .

The Orthogonal Projection onto $R(A)$

It can be shown that if A is $m \times n$ ($m \geq n$) and has **full rank**, then the orthogonal projection P_A onto $R(A)$ is given by

$$P_A = A(A^*A)^{-1}A^*$$

2.4 Some Special Matrices

2.4.1 Diagonal and Triangular Matrices

An $m \times n$ matrix $A = (a_{ij})$ is a **diagonal matrix** if $a_{ij} = 0$ for $i \neq j$. We write $A = \text{diag}(a_{11}, \dots, a_{ss})$, where $s = \min(m, n)$. An $n \times n$ matrix A is a **block diagonal matrix** if it is a diagonal matrix whose each diagonal entry is a square matrix. It is written as:

$$A = \text{diag}(A_{11}, \dots, A_{kk}),$$

where each A_{ii} is a square matrix. The sum of the orders of A_{ii} , $i = 1, \dots, k$ is n .

An $m \times n$ matrix $A = (a_{ij})$ is an **upper triangular matrix** if $a_{ij} = 0$ for $i > j$.

The transpose of an upper triangular matrix is **lower triangular**; that is, $A = (a_{ij})$ is lower triangular if $a_{ij} = 0$ for $i < j$.

Some Useful Properties of Triangular Matrices

The following properties of triangular matrices are useful.

- The product of two square upper (lower) triangular matrices is an upper (lower) triangular matrix. The diagonal entries of the product matrix are just the products of the diagonal entries of the individual matrices.
- The inverse of a nonsingular upper (lower) triangular matrix is a nonsingular upper (lower) triangular matrix. The diagonal entries of the inverse are the reciprocals of the diagonal entries of the original matrix.
- The eigenvalues of a square triangular matrix are its diagonal entries.
- The determinant of a square triangular matrix is the product of its diagonal entries.

2.4.2 Unitary (Orthogonal) Matrix

A complex square matrix U is **unitary** if $UU^* = U^*U = I$, where $U^* = (\overline{U})^T$. A real square matrix O is **orthogonal** if $OO^T = O^TO = I$. If U is an $n \times k$ matrix such that $U^*U = I_k$, then U is said to be **orthonormal**.

Orthogonal matrices play a very important role in numerical matrix computations.

The following important properties of orthogonal (unitary) matrices are attractive for numerical computations :

- The inverse of an orthogonal (unitary) matrix O is just its transpose (conjugate transpose).
- The product of two orthogonal (unitary) matrices is an orthogonal (unitary) matrix.
- The 2-norm and the Frobenius norm are invariant under multiplication by an orthogonal (unitary) matrix (**See Section 2.6**).
- The error in multiplying a matrix by an orthogonal matrix is not magnified by the process of numerical matrix multiplication (**See Chapter 3**).

2.4.3 Permutation Matrix

A nonzero square matrix P is called a **permutation matrix** if there is exactly one nonzero entry in each row and column which is 1 and the rest are all zero.

Effects of Pre-multiplication and Post-multiplication by a permutation matrix.

When a matrix A is premultiplied by a permutation matrix P , the effect is a permutation of the rows of A . Similarly, if A is postmultiplied by a permutation matrix, the effect is a permutation of the columns of A .

Some Important Properties of Permutation Matrices

- A permutation matrix is an orthogonal matrix
- The inverse of a permutation matrix P is its transpose and it is also a permutation matrix.
- The product of two permutation matrices is a permutation matrix.

2.4.4 Hessenberg (Almost Triangular) Matrix

A square matrix A is **upper Hessenberg** if $a_{ij} = 0$ for $i > j + 1$. The transpose of an upper Hessenberg matrix is a lower Hessenberg matrix, that is, a square matrix $A = (a_{ij})$ is a **lower Hessenberg matrix** if $a_{ij} = 0$ for $j > i + 1$. A square matrix A that is both upper and lower Hessenberg is **tridiagonal**.

$$\begin{array}{cc}
 \begin{pmatrix} * & * & & 0 \\ \vdots & & \ddots & \\ * & \vdots & & * \\ * & * & \dots & * \end{pmatrix} & \begin{pmatrix} * & \dots & * & * \\ * & \dots & * & * \\ \ddots & \vdots & \vdots & \vdots \\ 0 & & * & * \end{pmatrix} \\
 \text{Lower Hessenberg} & \text{Upper Hessenberg}
 \end{array}$$

An upper Hessenberg matrix $A = (a_{ij})$ is **unreduced** if

$$a_{i,i-1} \neq 0 \quad \text{for } i = 2, 3, \dots, n$$

Similarly, a lower Hessenberg matrix $A = (a_{ij})$ is **unreduced** if

$$a_{i,i+1} \neq 0 \quad \text{for } i = 1, 2, \dots, n-1$$

2.4.5 Companion Matrix

An unreduced upper Hessenberg matrix of the form

$$C = \begin{pmatrix} 0 & 0 & \dots & \dots & c_1 \\ 1 & 0 & \dots & \dots & c_2 \\ 0 & 1 & \dots & \dots & \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & c_n \end{pmatrix}$$

is called an **upper companion matrix**. The transpose of an upper companion matrix is a **lower companion matrix**.

The **characteristic polynomial** of the companion matrix C is:

$$\det(\lambda I - C) = \det(\lambda I - C^T) = \lambda^n - c_n \lambda^{n-1} - c_{n-1} \lambda^{n-2} - \dots - c_2 \lambda - c_1.$$

2.4.6 Nondiagonal Matrix

A matrix A is **nondiagonal** if and only if it is similar to a companion matrix of its characteristic polynomial. That is, A is a nondiagonal matrix if and only if there exists a nonsingular matrix T such that $T^{-1}AT$ is a companion matrix.

Remark: An unreduced Hessenberg matrix is nondiagonal, but the converse is not true.

2.4.7 The Jordan Canonical Form of a Matrix

For an $n \times n$ complex matrix A , there exists a nonsingular matrix T such that

$$T^{-1}AT = J = \text{diag}(J_1, \dots, J_k),$$

where

$$J_i = \begin{pmatrix} \lambda_i & 1 & & 0 \\ & \lambda_i & 1 & \\ & & \ddots & \ddots & \\ 0 & & & \ddots & 1 \\ & & & & \lambda_i \end{pmatrix}$$

is $m_i \times m_i$ and $m_1 + \dots + m_k = n$.

The matrices J_i are called **Jordan matrices** or **Jordan blocks** and J is called the **Jordan Canonical Form (JCF)** of A . For each $j = 1, 2, \dots, k$, λ_j is the eigenvalue of A with multiplicity m_j . **The same eigenvalue can appear in more than one blocks.**

Note: The matrix A is **nondiagonalizable** if its Jordan Canonical form has only one Jordan block associated with each distinct eigenvalue.

If $T = (t_1, t_2, \dots, t_{m_1}; t_{m_1+1}, \dots, t_{m_2}; \dots, t_n)$.

Then t_1, \dots, t_{m_1} must satisfy

$$At_1 = \lambda_1 t_1$$

and $At_{i+1} = \lambda_1 t_{i+1} + t_i, i = 1, 2, \dots, m_1 - 1$.

Similarly relations hold for the other vectors in T . The vectors t_i are called the **generalized eigenvectors** or **principal vectors** of A .

2.4.8 Positive Definite Matrix

A real symmetric matrix A is **positive definite** (positive semidefinite) if $x^T Ax > 0 (\geq 0)$ for every nonzero vector x .

Similarly, a complex Hermitian matrix A is positive definite (positive semidefinite) if $x^* Ax > 0 (\geq 0)$ for every nonzero complex vector x .

A commonly used notation for a symmetric positive definite (positive semidefinite) matrix is $A > 0 (\geq 0)$.

Unless otherwise mentioned a real symmetric or a complex Hermitian positive definite matrix will be referred to as a **positive definite matrix**.

Some Characterizations and Properties of Positive Definite Matrices

- A symmetric (Hermitian) matrix A is positive definite if and only if all its eigenvalues are positive.

- A symmetric (Hermitian) matrix A is positive definite if and only if all its leading principal minors are positive.

There are n leading principal minors of an $n \times n$ matrix A . The i^{th} leading principal minor is the determinant of the submatrix of A formed out of the first i rows and i columns.

- If $A = (a_{ij})$ is positive definite, then $a_{ii} > 0$ for all i .
- If $A = (a_{ij})$ is positive definite, then the largest element (in magnitude) of the whole matrix must lie on the diagonal.
- A symmetric positive definite matrix A admits the **Cholesky factorization** $A = HH^T$, where H is a lower triangular matrix with positive diagonal entries. **The most numerically efficient and stable way to check if a real symmetric matrix is positive definite is to compute its Cholesky factorization and see if the diagonal entries of the Cholesky factor are all positive.** See Chapter 3 (Section 3.4.2) for details.

2.4.9 Block Matrices

A matrix whose each entry is a matrix is called a **block matrix**. A **block diagonal matrix** is a diagonal matrix whose each entry is a matrix. A **block triangular matrix** is similarly defined.

The Jordan Canonical form is an example of a **block diagonal matrix**.

The following are easily verified properties of the 2×2 block triangular matrix A .

- $\begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}^{-1} = \begin{pmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12}A_{22}^{-1} \\ 0 & A_{22}^{-1} \end{pmatrix}$,

assuming that A_{11} and A_{22} are nonsingular.

- $\det(A) = \det(A_{11}) \cdot \det(A_{22})$.
- $\det(\lambda I - A) = \det(\lambda I - A_{11}) \cdot \det(\lambda I - A_{22})$

In general, suppose A is partitioned in the form

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

then

- A is nonsingular if and only if $A_S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is nonsingular (assuming that A_{11} is nonsingular) and in this case, the inverse of A is given by:

$$A^{-1} = \begin{pmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}A_S^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}A_S^{-1} \\ -A_S^{-1}A_{21}A_{11}^{-1} & A_S^{-1} \end{pmatrix}.$$

- A is nonsingular if and only if $\hat{A}_S = A_{11} - A_{12}A_{22}^{-1}A_{21}$ is nonsingular (assuming that A_{22} is nonsingular), and the inverse of A is given by:

$$A^{-1} = \begin{pmatrix} \hat{A}_S^{-1} & -\hat{A}_S^{-1}A_{12}A_{22}^{-1} \\ -A_{22}^{-1}A_{21}\hat{A}_S^{-1} & A_{22}^{-1} + A_{22}^{-1} + A_{22}^{-1}A_{21}\hat{A}_S^{-1}A_{12}A_{22}^{-1} \end{pmatrix}.$$

- $\det(A) = \det(A_{22}) \cdot \det(\hat{A}_S)$.

The matrices A_S and \hat{A}_S are called the **Schur Complements** of A with respect to A_{11} and A_{22} , respectively.

2.5 Vector and Matrix Norms

2.5.1 Vector Norms

Let

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

be an n -vector in \mathbb{C}^n . Then, a vector norm, denoted by the symbol $\|x\|$, is a real-valued **continuous** function of the components x_1, x_2, \dots, x_n of x , satisfying the following properties:

1. $\|x\| > 0$ for every non-zero x . $\|x\| = 0$ if and only if x is the zero vector.
2. $\|\alpha x\| = |\alpha| \|x\|$ for all x in \mathbb{C}^n and for all scalars α .
3. $\|x + y\| \leq \|x\| + \|y\|$ for all x and y in \mathbb{C}^n .

The last property is known as the **Triangle Inequality**.

Note: $\|-x\| = \|x\|$ and $|\|x\| - \|y\|| \leq \|x - y\|$. It is simple to verify that the following are vector norms.

Some Commonly Used Vector Norms

1. $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$ (sum norm or 1-norm)
2. $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ (Euclidean norm or 2-norm)
3. $\|x\|_\infty = \max_i |x_i|$ (maximum or ∞ -norm)

The above three are special case of the **p -norm** or **Hölder norm** defined by $\|x\|_p = (|x_1|^p + \cdots + |x_n|^p)^{\frac{1}{p}}$ for any $p \geq 1$.

Unless otherwise stated, by $\|x\|$ we will mean $\|x\|_2$.

Example 2.5.1

Let $x = (1, 1, -2)^T$. Then $\|x\|_1 = 4$, $\|x\|_2 = \sqrt{1^2 + 1^2 + (-2)^2} = \sqrt{6}$, and $\|x\|_\infty = 2$.

An important property of the Hölder norm is the **Hölder inequality**

$$|x^*y| \leq \|x\|_p \|y\|_q, \quad \frac{1}{p} + \frac{1}{q} = 1.$$

A special case of the Hölder-inequality is the **Cauchy-Schwartz** inequality: $|x^*y| \leq \|x\|_2 \|y\|_2$,

Equivalence Property of the Vector norms

All vector norms are **equivalent** in the sense that there exist positive constants α and β such that $\alpha\|x\|_\mu \leq \|x\|_\nu \leq \beta\|x\|_\mu$, for all x , where μ and ν specify the nature of norms.

For the 2, 1, or ∞ norms, we can compute α and β easily and have the following inequalities:

Theorem 2.5.1 *Let x be in \mathbb{C}^n . Then*

1. $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$
2. $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$
3. $\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty$

2.5.2 Matrix Norms

Let A be an $m \times n$ matrix. Then, analogous to the vector norm, we define the matrix norm for $\|A\|$ in $\mathbb{C}^{m \times n}$ with the following properties:

1. $\|A\| \geq 0$; $\|A\| = 0$ if and only if A is the zero matrix
2. $\|\alpha A\| = |\alpha| \|A\|$ for any scalar α
3. $\|A + B\| \leq \|A\| + \|B\|$

Subordinate Matrix Norms

Given a matrix A and a vector norm $\|\cdot\|_p$ on \mathbb{C}^n , a non-negative number defined by:

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

satisfies all the properties of a matrix norm. This norm is called the matrix norm **subordinate** to (or induced by) the p -norm.

A very useful and frequently used property of a subordinate matrix norm $\|A\|_p$ (**we shall sometimes call it the p -norm of a matrix A**) is

$$\|Ax\|_p \leq \|A\|_p \|x\|_p.$$

Let A be an $m \times n$ matrix. Then

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|$$

(maximum column sum norm)

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|$$

(maximum row sum norm)

Example 2.5.2

$$A = \begin{pmatrix} 1 & -2 \\ 3 & 4 \\ -5 & 6 \end{pmatrix}$$

$$\|A\|_1 = 12$$

$$\|A\|_\infty = 11$$

Another useful p -norm is the **spectral norm**.

Definition 2.5.1 The spectral norm a matrix A is $\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$.

It can be shown that $\|A\|_2 = \sqrt{\text{largest eigenvalue of } A^*A}$

(Note that the eigenvalues of A^*A are real and non-negative).

Example 2.5.3

$$A = \begin{pmatrix} 2 & 5 \\ 1 & 3 \end{pmatrix}$$

$$A^T A = \begin{pmatrix} 5 & 13 \\ 13 & 34 \end{pmatrix}$$

The eigenvalues of $A^T A$ are 0.0257 and 38.9743.

$$\|A\|_2 = \sqrt{38.9743} = 6.2429$$

The Frobenius Norm

An important matrix norm is the Frobenius norm:

$$\|A\|_F = \left[\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right]^{\frac{1}{2}}$$

A matrix norm $\|\cdot\|_M$ and a vector norm $\|\cdot\|_v$ are consistent if for all matrices A and vectors x , the following inequality holds:

$$\|Ax\|_v \leq \|A\|_M \|x\|_v.$$

Consistency Property of the Matrix Norm

A matrix norm is consistent if, for any two matrices A and B , compatible for matrix multiplication, the following property is satisfied:

$$\|AB\| \leq \|A\| \|B\|.$$

The Frobenius norm and all subordinate norms are consistent.

Notes:

1. For the identity matrix I , $\|I\|_F = \sqrt{n}$, whereas $\|I\|_1 = \|I\|_2 = \|I\|_\infty = 1$.
2. $\|A\|_F^2 = \text{trace}(A^*A)$, where trace (A) is defined as the sum of the diagonal entries of A , that is, if $A = (a_{ij})$, then $\text{trace}(A) = a_{11} + a_{22} + \dots + a_{nn}$. The trace of A will, sometimes, be denoted by $Tr(A)$ or $tr(A)$.

Equivalence Property of Matrix Norms

As in the case of vector norms, the matrix norms are also related. There exist scalars α and β such that: $\alpha\|A\|_\mu \leq \|A\|_\nu \leq \beta\|A\|_\mu$. In particular, the following inequalities relating various matrix norms are true and are used very frequently in practice. We state the Theorem without proof. For a proof, see Datta (1995, pp 28-30).

Theorem 2.5.2 Let A be $m \times n$. Then

- (1) $\frac{1}{\sqrt{n}}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{m}\|A\|_\infty$.
- (2) $\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2$.
- (3) $\frac{1}{\sqrt{m}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1$.
- (4) $\|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty}$.

2.6 Norm Invariant Properties Under Unitary Matrix Multiplication

We conclude the chapter by listing some very useful norm properties of unitary matrices that are often used in practice.

Theorem 2.6.1 Let U be an unitary matrix. Then

$$\|U\|_2 = 1.$$

Proof: $\|U\|_2 = \sqrt{\rho(U^*U)} = \sqrt{\rho(I)} = 1$. (Recall that $\rho(A)$ denotes the spectral radius of A).

The next two theorems show that **2-norm and the Frobenius norm are invariant under multiplication by a unitary matrix.**

Theorem 2.6.2 Let U be an unitary matrix and AU be defined. Then

$$(i) \|AU\|_2 = \|A\|_2$$

$$(ii) \|AU\|_F = \|A\|_F$$

Proof:

$$(i) \|AU\|_2 = \sqrt{\rho(U^*A^*AU)} = \sqrt{\rho(A^*A)} = \|A\|_2$$

$$(ii) \|AU\|_F = \text{trace}(U^*A^*AU) = \text{trace}(A^*A) = \|A\|_F^2$$

Thus $\|AU\|_F = \|A\|_F$. ■

Similarly, if UA is defined, then we have

Theorem 2.6.3 (i) $\|UA\|_2 = \|A\|_2$

$$(ii) \|UA\|_F = \|A\|_F$$

Proof: The proof is similar to Theorem 2.6.2.

2.7 Kronecker Product, Kronecker Sum and Vec Operation

Let $A \in \mathbb{C}^{m \times n}$ and $B \in \mathbb{C}^{r \times s}$, then the $mr \times ns$ matrix defined by

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}$$

is called the **Kronecker product** of A and B .

If A and B are invertible, then $A \otimes B$ is invertible and $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.

The Eigenvalues of the Kronecker Product and Kronecker Sum

Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of $A \in \mathbb{C}^{n \times n}$, and μ_1, \dots, μ_m be the eigenvalues of $B \in \mathbb{C}^{m \times m}$. Then it can be shown [Exercise 38] that the eigenvalues of $A \otimes B$ are the mn numbers $\lambda_i \mu_j, i = 1, \dots, n; j = 1, \dots, m$, and the eigenvalues of $A \oplus B$ are the mn numbers $\lambda_i + \mu_j, i = 1, \dots, n; j = 1, \dots, m$.

vec Operation

Let $X \in \mathbb{C}^{m \times n}$ and $X = (x_{ij})$.

Then the vector obtained by stacking the columns of X in one vector is denoted by $\text{vec}(X)$:

$$\text{vec}(X) = (x_{11}, \dots, x_{m1}, x_{12}, \dots, x_{m2}, \dots, x_{1n}, \dots, x_{mn})^T$$

If $A \in \mathbb{C}^{m \times m}$ and $B \in \mathbb{C}^{n \times n}$, then it can be shown [Exercise 39] that $\text{vec}(AX + XB) = ((I_n \otimes A) + (B^T \otimes I_m))\text{vec } X$.

The Kronecker products and vec operations are useful in the study of the existence, uniqueness, sensitivity, and numerical solutions of the Lyapunov and Sylvester equations (see Chapter 8 and Chapter 15).

REVIEW EXERCISES

1. Prove that
 - (a) a set of m vectors in \mathbb{R}^n , where $m > n$, is linearly dependent.
 - (b) $\text{span}\{v_1, \dots, v_k\}$ is a subspace of \mathbb{R}^n , where $\text{span}\{v_1, \dots, v_k\}$ is the set of linear combinations of the k vectors v_1, \dots, v_k from a vector space \mathbb{R} .
 - (c) $\text{span}\{v_1, \dots, v_k\}$ is the smallest subspace of \mathbb{R}^n containing v_1, \dots, v_k .
2. Prove that if $S = \{s, \dots, s_k\}$ is an orthogonal set of nonzero vectors, then S is linearly independent.
3. Let S be an m dimensional subspace of \mathbb{R}^n . Then prove that S has an orthonormal basis.
(Hint: Let $S = \{v_1, \dots, v_m\}$.) Define a set of vectors $\{u_k\}$ by:

$$u_1 = \frac{v_1}{\|v_1\|} \text{ and } u_{k+1} = \frac{v'_{k+1}}{\|v'_{k+1}\|},$$

where

$$v'_{k+1} = v_{k+1} - (v_{k+1}^T u_1)u_1 - (v_{k+1}^T u_2)u_2 - \cdots - (v_{k+1}^T u_k)u_k, \quad k = 1, 2, \dots, m-1.$$

Then show that $\{u_1, \dots, u_m\}$ is an orthonormal basis of S .

This is the classical **Gram-Schmidt process**.

4. Using the Gram-Schmidt process construct an orthonormal basis of \mathbb{R}^3 .
5. Let S_1 and S_2 be two subspaces of \mathbb{R}^n . Then prove that

$$\dim(S_1 + S_2) = \dim(S_1) + \dim(S_2) - \dim(S_1 \cap S_2).$$

6. Prove the following basic facts on nonsingularity and the inverse of A :

- (a) $(A^{-1})^{-1} = A$
- (b) $(A^T)^{-1} = (A^{-1})^T$
- (c) $(cA)^{-1} = \frac{1}{c}A^{-1}$, where c is a nonzero scalar
- (d) $(AB)^{-1} = B^{-1}A^{-1}$
7. Let $A = \begin{pmatrix} A_1 & 0 \\ A_2 & A_3 \end{pmatrix}$, where A_1 and A_3 are square. Prove that $\det(A) = \det(A_1)\det(A_3)$.
8. (a) Show that if P_S is an orthogonal projection onto S , then $I - P_S$ is the orthogonal projection onto S^\perp .
- (b) Prove that
 - i. $P_A = A(A^T A)^{-1}A^T$.

- ii. $P_N = I - A(A^T A)^{-1}A^T$.
- (c) Let b_R and b_N be, respectively, the orthogonal projections of the vector b onto $R(A)$ and $N(A)$. Then
- $b_R = P_A b$
 - $b_N = P_N b$
9. (a) Find P_A and P_N for the matrices
- $$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 0 & 0 \end{pmatrix}, \text{ and } A = \begin{pmatrix} 1 & 1 \\ 10^{-4} & 0 \\ 0 & 10^{-4} \end{pmatrix}$$
- (b) For the vector $b = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$, find b_R and b_N for each of the above matrices.
- (c) Find an orthonormal basis for each of the above matrices using the Gram-Schmidt process and then find P_A , P_N , b_R , and b_N .
10. (**Distance between two subspaces**). Let S_1 and S_2 be two subspaces of \mathbb{R}^n such that $\dim(S_1) = \dim(S_2)$. Let P_1 and P_2 be the orthogonal projections onto S_1 and S_2 respectively. Then $\|P_1 - P_2\|_2$ is defined to be the distance between S_1 and S_2 . Prove that the distance between S_1 and S_2 , $\text{dist}(S_1, S_2) = (\theta)$, where θ is the angle between S_1 and S_2 , that is the minimal possible angle between x in S_1 and y in S_2 .
11. Prove that if P_S is an orthogonal projection onto S , then $I - 2P_S$ is an orthogonal projection.
12. Prove the following:
- The product of two square upper (lower) triangular matrices is an upper (lower) triangular matrix . In general, if $A = (a_{ij})$ is an upper (lower) triangular matrix , then $p(A)$, where $p(x)$ is a polynomial, is an upper (lower) triangular matrix whose diagonal elements are $p(a_{ii})$, $i = 1, \dots, n$.
 - The inverse of a lower (upper) triangular matrix is another lower (upper) triangular matrix, whose diagonal entries are the reciprocals of the diagonal entries of the triangular matrix.
 - The determinant of a triangular matrix is the product of its diagonal entries.
 - The eigenvalues of a triangular matrix are its diagonal entries.
 - The inverse of a nonsingular matrix A can be written as a polynomial in A . (Hint: use Cayley-Hamilton theorem).
13. Prove that the product of an upper Hessenberg matrix and a square upper triangular matrix is an upper Hessenberg matrix.

14. Prove that a symmetric Hessenberg matrix is symmetric tridiagonal.
15. (a) Prove that the product of two orthogonal matrices is an orthogonal matrix.
 (b) Prove that a triangular matrix that is orthogonal is diagonal.
16. Let $A = (a_{ij})$ be an $n \times n$ symmetric positive definite matrix. Prove the following.
- (a) Each diagonal entry of A must be positive.
 - (b) A is nonsingular.
 - (c) $(a_{ij})^2 < a_{ii}a_{jj}$ for $i = 1, \dots, n, j = 1, \dots, n, i \neq j$
 - (d) The largest element of the matrix must lie on the diagonal.
17. Let A be a symmetric positive definite matrix and x be a nonzero n -vector. Prove that $A + xx^T$ is positive definite.
18. Prove that if the eigenvalues of a matrix A are all distinct, then A is nonderogatory.
19. Prove that a symmetric matrix A is positive definite if and only if A^{-1} exists and is positive definite.
20. Let A be an $m \times n$ matrix ($m \geq n$) having full rank. Then $A^T A$ is positive definite.
21. Prove the following basic facts on the eigenvalues and eigenvectors:
- (a) A matrix A is nonsingular if and only if A does not have a zero eigenvalue. (Hint: $\det A = \lambda_1\lambda_2, \dots, \lambda_n$.)
 - (b) The eigenvalues of A^T and A are the same.
 - (c) Two similar matrices have the same eigenvalues, but the converse is not necessarily true.
 - (d) A symmetric matrix is positive definite if and only if all its eigenvalues are positive.
 - (e) The eigenvalues of a unitary (orthogonal) matrix have moduli 1.
 - (f) The eigenvectors of a symmetric matrix can be chosen to be orthogonal.
 - (g) Let A be a symmetric matrix and let Q be orthogonal such that QAQ^T is diagonal. Then show that the columns of Q are the eigenvectors of A .
22. Let $\text{trace}(A) = \sum_{i=1}^n a_{ii}$. Then prove the following:
- (a) $\text{trace}(AB) = \text{trace}(BA)$.
 - (b) $\text{trace}(AA^*) = \sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2$, where $A = (a_{ij})$ is $m \times n$.
 - (c) $\text{trace}(A + B) = \text{trace}(A) + \text{trace}(B)$.

(d) $\text{trace}(TAT^{-1}) = \text{trace}(A)$.

23. Show that $\|x\|_1$, $\|x\|_\infty$, $\|x\|_2$ are vector norms.

24. Show that, if x and y are two vectors, then

$$\left| \|x\| - \|y\| \right| \leq \|x - y\| \leq \|x\| + \|y\|$$

25. If x and y are two n -vectors, then prove that

(a) $|x^T y| \leq \|x\|_2 \|y\|_2$ (**Cauchy-Schwarz** inequality)

(b) $\|xy^T\|_2 \leq \|x\|_2 \|y\|_2$ (**Schwarz** inequality)

26. Let x and y be two orthogonal vectors, then prove that

$$\|x + y\|_2^2 = \|x\|_2^2 + \|y\|_2^2.$$

27. Prove that for any vector x , we have

$$\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1.$$

28. Prove that $\|A\|_1$, $\|A\|_2$, and $\|A\|_\infty$ are matrix norms.

29. Let $A = (a_{ij})$ be $m \times n$. Define $A_\ell = \max_{ij} |a_{ij}|$. Is A_ℓ a matrix norm? Give reasons for your answer.

30. (a) Prove that the vector length is preserved by orthogonal matrix multiplication. That is, if $x \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n \times n}$ is orthogonal, then $\|Qx\|_2 = \|x\|_2$ (**Isometry Lemma**).

(b) Is the statement in part (a) true if the 1-norm and ∞ -norm are used? Give reasons. What if the Frobenius norm is used? (**Hint:** Use Theorems 2.6.2 and 2.6.3).

31. Prove that for any $n \times m$ unitary matrix U , $\|U\|_2 = 1$ and $\|U\|_F = \sqrt{m}$

32. Prove that if Q and P are unitary matrices, then

(a) $\|QAP\|_F = \|A\|_F$

(b) $\|QAP\|_2 = \|A\|_2$

33. Prove that the spectral norm of a Hermitian matrix is the same as its spectral radius.

34. Prove that $\|A\|_2 \leq \|A\|_F$ (**Hint:** Use Cauchy-Schwarz inequality).

35. Prove that

$$\|AB\|_F \leq \|A\|_F \|B\|_2.$$

$$\|AB\|_F \leq \|A\|_2 \|B\|_F.$$

36. Prove that $\|A^T A\|_2 = \|A\|_2^2$.

37. Let $A = (a_1, \dots, a_n)$, where a_j is the j^{th} column of A . Then prove that

$$\|A\|_F^2 = \sum_{i=1}^n \|a_i\|_F^2.$$

38. Let $A \in \mathbb{C}^{n \times n}, B \in \mathbb{C}^{m \times m}$. Let $\lambda_1, \lambda_2, \dots, \lambda_n$ and μ_1, \dots, μ_m be the respective eigenvalues. Then prove that

- (a) The eigenvalues of $A \otimes B$ are the mn numbers $\lambda_i \mu_j, i = 1, \dots, n; j = 1, \dots, m$, where $\lambda_1, \lambda_2, \dots, \lambda_n$ and $\mu_1, \mu_2, \dots, \mu_m$ are the eigenvalues of A and B , respectively.
- (b) The eigenvalues of $A \oplus B$ are the mn numbers $\lambda_i + \mu_j, i = 1, 2, \dots, n; j = 1, 2, \dots, m$.

39. If $A \in \mathbb{C}^{m \times m}$ and $B \in \mathbb{C}^{n \times n}$, then prove that

$$\text{vec } (AX + XB) = ((I_n \otimes A) + (B^T \otimes I_m)) \text{ vec } X.$$

40. Prove that if A and B are invertible, then $A \otimes B$ is invertible and

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}.$$

References

1. H. Anton and C. Rorres, *Elementary Linear Algebra with Applications*, John Wiley, New York, 1987.
2. R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
3. D.C. Lay, *Linear Algebra and Its Applications*, Addison-Wesley, Reading, MA, 1994.
4. S.J. Leon, *Linear Algebra with Applications*, Macmillan, New York, fourth edition, 1994.
5. B. Noble and J.W. Daniel, *Applied Linear Algebra*, Prentice-Hall, Englewood Cliffs, NJ, 1977.
6. G. Strang, *Linear Algebra and Its Applications*, 3rd Edition, Academic Press, New York, 1988.

Chapter 3

SOME FUNDAMENTAL TOOLS AND CONCEPTS FROM NUMERICAL LINEAR ALGEBRA

Contents

3.1	Introduction	61
3.2	Floating Point Numbers and Errors in Computations	61
3.2.1	Floating Point Numbers	61
3.2.2	Rounding Errors	62
3.2.3	Laws of Floating Point Arithmetic	63
3.2.4	Catastrophic Cancellation	64
3.3	Conditioning, Efficiency, Stability and Accuracy	64
3.3.1	Algorithms and Pseudocodes	64
3.3.2	Solving an Upper Triangular System	65
3.3.3	Solving a Lower Triangular System	65
3.3.4	Efficiency of an Algorithm	66
3.3.5	The Concept of Numerical Stability	66
3.3.6	Conditioning of the Problem and Perturbation Analysis	68
3.3.7	Conditioning of the Problem, Stability of the Algorithm, and Accuracy of the Solution	69
3.3.8	Conditioning of the Linear System and Eigenvalue Problems	69
3.4	LU Factorization	73
3.4.1	LU Factorization using Gaussian Elimination	73
3.4.2	The Cholesky Factorization	78
3.4.3	LU Factorization of an Upper Hessenberg Matrix	79
3.5	Numerical Solution of the Linear System $Ax = b$.	81

3.5.1	Solving $Ax = b$ using the inverse of A	81
3.5.2	Solving $Ax = b$ using Gaussian Elimination with Partial Pivoting	81
3.5.3	Solving a Hessenberg Linear System	83
3.5.4	Solving $AX = B$.	83
3.5.5	Finding the Inverse of A	83
3.5.6	Computing the Determinant of A	83
3.5.7	Iterative Refinement	84
3.6	The QR Factorization	84
3.6.1	Householder Matrices	85
3.6.2	The Householder QR Factorization	85
3.6.3	Givens Matrices	88
3.6.4	The QR Factorization Using Givens Rotations	90
3.6.5	The QR Factorization of a Hessenberg Matrix Using Givens Matrices	91
3.7	Orthonormal Bases and Orthogonal Projections Using QR Factorization	91
3.8	The Least-Squares Problem	92
3.8.1	Solving the Least-Squares Problem Using Normal Equations	92
3.8.2	Solving the Least-Squares Problem Using QR Factorization	93
3.9	The Singular Value Decomposition (SVD)	94
3.9.1	The Singular Value Decomposition and the Structure of a Matrix	96
3.9.2	Orthonormal Bases and Orthogonal Projections	97
3.9.3	The Rank and the Rank-Deficiency of a Matrix	98
3.9.4	Numerical Rank	99
3.9.5	Solving the Least Squares Problem Using the Singular Value Decomposition	100
3.10	Summary and Review	102
3.11	Chapter Notes and Further Reading	105

Topics Covered

- Floating Point Numbers and Errors in Computations
- *LU* Factorization Using Gaussian Elimination
- *QR* Factorization Using Householder and Givens Matrices
- Numerical Solution of the Algebraic Linear Systems and Least-Squares Problems
- The Singular Value Decomposition (SVD).

3.1 Introduction

In this chapter, we introduce some fundamental concepts and techniques of numerical linear algebra which, we believe, are essential for in-depth understanding of computational algorithms for control problems, discussed in this book. The basic concepts of floating point operations, numerical stability of an algorithm, conditioning of a computational problem and their effects on the accuracy of a solution obtained by a certain algorithm are introduced first.

Three important matrix factorizations: **LU** , **QR** , and the **Singular Value Decomposition** (SVD), and their applications to solutions of algebraic linear systems, linear least-squares problems and eigenvalue problems are next described in details.

The method of choice for the linear system problem is the **LU factorization technique obtained by Gaussian elimination with partial pivoting** (Section 3.4). The method of choice for the symmetric positive definite system is the **Cholesky factorization technique** (Algorithm 3.4.1).

The **QR** factorization of a matrix is introduced in the context of the least-squares solution of a linear system; however, it also forms the core of the **QR iteration technique, which is the method of choice for eigenvalue computation**. The **QR iteration technique** itself for eigenvalue computation is described in Chapter 4. Two numerically stable methods for the **QR factorization**; namely, **Householder's and Givens' methods are described in Section 3.6**. Householder's method is slightly cheaper than Givens' method for sequential computations, but the latter has computational advantages in parallel computation setting.

The Singular Value Decomposition (SVD) has become nowadays an essential tool for determining the **numerical rank**, the **distance of a matrix from a matrix of immediate lower rank**, **finding the orthonormal basis and projections**, etc. This important matrix factorization is described in Section 3.9. The SVD is also a reliable tool for computing the least-squares solution to $Ax = b$.

A reliable and widely-used computational technique for computing the SVD of a matrix is described in Chapter 4.

3.2 Floating Point Numbers and Errors in Computations

3.2.1 Floating Point Numbers

Most scientific and engineering computations on a computer are performed using **floating-point arithmetic**. Computers may have different bases, though base 2 is most common.

A t -digit **floating point** number in base β has the form:

$$x = m \cdot \beta^e,$$

where m is a t -digit fraction, called **mantissa**, and e is called **exponent**.

If the first digit of the mantissa is different from zero, then the floating point number is called **normalized**. Thus 0.3457×10^5 is a 4-digit normalized decimal floating number, whereas

0.03475×10^6 is a five-digit unnormalized decimal floating point number.

The number of digits in the mantissa is called the **precision**. On many computers, it is possible to manipulate floating point numbers so that a number can be represented with about twice the usual precision. Such a precision is called **double precision**.

Most computers nowadays conform to the IEEE floating point standard (ANSI/IEEE standard 754-1985). For a single-precision, IEEE standard recommends about 24 binary digits and for a double precision, about 53 binary digits. Thus, IEEE standard for **single precision provides approximately seven decimal digits of accuracy**, since $2^{-23} \cong 1.2 \times 10^{-7}$, and **double precision provides approximately sixteen decimal digits of accuracy**, since $2^{-52} \approx 2.2 \times 10^{-16}$.

Note: Although computations with double precision increase accuracy, they require more computer time and storage.

On each computer, there is an allowable range of the exponent $e : L$, the minimum; U , the maximum. **L and U vary from computer to computer.**

If, during computations, the computer produces a number whose exponent is too large (too small), that is, it is outside the permissible range, then we say that an **overflow** (**underflow**) has occurred.

Overflow is a serious problem; for most systems, the result of an overflow is $\pm\infty$. Underflow is usually considered less serious. On most computers, when an underflow occurs, the computed value is set to zero, and then computations proceed. **Unless otherwise stated, we will use only decimal arithmetic.**

3.2.2 Rounding Errors

If a computed result of a given real number is not machine representable, then there are two ways it can be represented in the machine. Consider the machine representation of the number

$$\pm \cdot d_1 \cdots d_t d_{t+1} \cdots$$

Then the first method, **chopping**, is the method in which the digits from d_{t+1} on are simply chopped off. The second method is **rounding**, in which the digits d_{t+1} through the rest are not only chopped off, but the digit d_t is also rounded up or down depending on whether $d_{t+1} \geq 5$ or $d_{t+1} < 5$.

We will denote the floating point representation of a real number x by $\text{fl}(x)$.

Example 3.2.1 Rounding

Let $x = 3.141596$

$$t = 2 : \text{fl}(x) = 3.1$$

$$t = 3 : \text{fl}(x) = 3.14$$

$$t = 4 : \text{fl}(x) = 3.142$$

A useful measure of error in computation is the **relative error**.

Definition 3.2.1 Let \hat{x} denote an approximation of x . Then the relative error is $\frac{|\hat{x} - x|}{|x|}$, $x \neq 0$.

We now give an expression for the relative error in representing a real number x by its floating point representation $\text{fl}(x)$. Proof of Theorem 3.2.1 can be found in Datta (1995, pp. 47).

Theorem 3.2.1 Let $\text{fl}(x)$ denote the floating point representation of a real number x in base β . Then

$$\frac{|\text{fl}(x) - x|}{|x|} \leq \mu = \begin{cases} \frac{1}{2}\beta^{1-t} & \text{for rounding} \\ \beta^{1-t} & \text{for chopping} \end{cases} \quad (3.2.1)$$

■

Definition 3.2.2 The number μ in the above theorem is called the **machine precision, computer epsilon or unit roundoff error**. It is the smallest positive floating point number such that

$$\text{fl}(1 + \mu) > 1.$$

The number μ is usually of order 10^{-16} and 10^{-7} (on most machines) for double and single precisions computations, respectively. For example, for the IBM 360 and 370, $\beta = 16$, $t = 6$, $\mu = 4.77 \times 10^{-7}$.

Definition 3.2.3 The **significant digits** in a number are the number of digits starting with the first nonzero digit.

For example, the number 1.5211 has five significant digits; whereas the number 0.0231 has only three.

3.2.3 Laws of Floating Point Arithmetic

The formula (3.2.1) can be written as

$$\begin{aligned} \text{fl}(x) &= x(1 + \delta) \\ \text{where } |\delta| &\leq \mu \end{aligned}$$

Assuming that the IEEE standard holds, we can easily derive the following simple **laws of floating point arithmetic**.

Theorem 3.2.2 (Laws of Floating Point Arithmetic). Let x and y be two floating point numbers, and let $\text{fl}(x + y)$, $\text{fl}(x - y)$, $\text{fl}(xy)$, and $\text{fl}(x/y)$ denote, respectively, the computed sum, difference, product and quotient. Then

1. $\text{fl}(x \pm y) = (x \pm y)(1 + \delta)$, where $|\delta| \leq \mu$.
2. $\text{fl}(xy) = (xy)(1 + \delta)$, where $|\delta| \leq \mu$.

3. if $y \neq 0$, then $\text{fl}(x/y) = (x/y)(1 + \delta)$, where $|\delta| \leq \mu$.

■

On computers that do not use the IEEE standard, the following floating point law of addition might hold:

4. $\text{fl}(x + y) = x(1 + \delta_1) + y(1 + \delta_2)$ where $|\delta_1| \leq \mu$, and $|\delta_2| \leq \mu$.

Example 3.2.2 Let $\beta = 10$, $t = 4$

$$\begin{aligned} x &= 0.1112, y = 0.2245 \times 10^5 \\ xy &= 0.24964 \times 10^4, \\ \text{fl}(xy) &= 0.24960 \times 10^4 \end{aligned}$$

Thus, $|\text{fl}(xy) - xy| = 0.4000$ and $|\delta| = 1.7625 \times 10^{-4} < \frac{1}{2} \times 10^{-3}$.

3.2.4 Catastrophic Cancellation

A phenomenon, called **catastrophic cancellation**, occurs when two numbers of approximately the same size are subtracted. Very often significant digits are lost in the process.

Consider the example of computing $f(x) = e^x - 1 - x$ for $x = 0.01$. In five digit arithmetic $a = e^x - 1 = 1.0101 - 1 = 0.0101$. Then the computed value of $f(x) = a - x = 0.0001$; whereas the true answer is 0.000050167.

Note that even though the subtraction was done accurately, the final result was wrong. **Fortunately, often cancellation can be avoided by rearranging computations.** For the example under consideration, if e^x were computed using the convergent series $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$, then the result would have been 0.000050167, which is correct up to five significant digits.

For details and examples, see Datta (1995, pp. 43-61). See also Stewart (1998, pp. 136-138) for an illuminating discussion on this topic.

3.3 Conditioning, Efficiency, Stability and Accuracy

3.3.1 Algorithms and Pseudocodes

Definition 3.3.1 An **algorithm** is an ordered set of operations, logical and arithmetic, which when applied to a computational problem defined by a given set of data, called the **input data**, produces a solution to the problem. A solution comprises of a set of data called the **output data**.

In this book, for the sake of convenience and simplicity, we will very often describe algorithms by means of **pseudocodes**, which can easily be translated into computer codes. Here are two illustrations.

3.3.2 Solving an Upper Triangular System

Consider the system

$$Ty = b$$

where $T = (t_{ij})$ is a nonsingular upper triangular matrix and $y = (y_1, y_2, \dots, y_n)^T$ and $b = (b_1, \dots, b_n)^T$.

Algorithm 3.3.1 *Back Substitution Method for Upper Triangular System*

Input: T – An $n \times n$ nonsingular upper triangular matrix
 b - An n -vector.

Output: The vector $y = (y_1, \dots, y_n)^T$ such that $Ty = b$.

Step 1. Compute $y_n = \frac{b_n}{t_{nn}}$

Step 2. For $i = n-1, n-2, \dots, 2, 1$ do

$$y_i = \frac{1}{t_{ii}} \left(b_i - \sum_{j=i+1}^n t_{ij}y_j \right)$$

End

Note: When $i = n$, the summation (\sum) is skipped.

3.3.3 Solving a Lower Triangular System

A lower triangular system can be solved in an analogous manner. The process is known as the **forward substitution method**. Let $L = (l_{ij})$, and $b = (b_1, b_2, \dots, b_n)^T$.

Algorithm 3.3.2 *The Forward Substitution Method for Lower Triangular System*

Inputs: L – A lower triangular matrix
 b - An n -vector

Output: y – An n -vector such that $Ly = b$.

Step 1. Compute $y_1 = \frac{b_1}{l_{11}}$

Step 2. For $i = 2, 3, \dots, n$. do

$$y_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij}y_j \right)$$

End

3.3.4 Efficiency of an Algorithm

Two most desirable properties for an algorithm are: **Efficiency and Stability**.

The efficiency of an algorithm is measured by the amount of computer time consumed in its implementation.

A theoretical and crude measure of efficiency is the number of floating point operations (**flops**) needed to implement the algorithm.

Definition 3.3.2 *A floating point operation or flop is a floating point operation: +, -, *, or /.*

Flop-Count for Algorithm 3.3.1 and Algorithm 3.3.2. Each of these algorithms requires n^2 flops.

The Big O Notation

An algorithm will be called an $O(n^p)$ algorithm if the dominant term in the operations count of the algorithm is a multiple of n^p . Thus, the solution of a triangular system is an $O(n^2)$ algorithm.

Notation for Overwriting and Interchange

We will use the notation

$$a \equiv b$$

to denote that “**b overwrites a**”. Similarly, if two computed quantities a and b are interchanged, they will be written symbolically

$$a \leftrightarrow b.$$

3.3.5 The Concept of Numerical Stability

The accuracy or the inaccuracy of the computed solution of a problem usually depends upon two important factors: **the stability or the instability of the algorithm used to solve the problem and the conditioning of the problem** (that is, how sensitive the problem is to small perturbations).

We first define the concept of stability of an algorithm. In the next section, we shall talk about the conditioning of a problem.

The study of stability of an algorithm is done by means of **roundoff error analysis**. There are two types: **backward error analysis** and **forward error analysis**.

In forward analysis an attempt is made to see how the computed solution obtained by the algorithm differs from the exact solution based on the same data.

On the other hand, backward analysis relates the error to the data of the problem rather than to the problem’s solution.

Definition 3.3.3 An algorithm is called **backward stable** if it produces an exact solution to a **nearby** problem; that is, a backward algorithm solves a problem whose data are close to the original data.

Backward error analysis, popularized in the literature by J. H. Wilkinson (1965), is now widely used in matrix computations and using this analysis, the stability (or instability) of many algorithms in numerical linear algebra has been established in recent years. **In this book, by “stability” we will imply “backward stability,” unless otherwise stated.**

As a simple example of backward stability, consider again the problem of computing the sum of two floating point numbers x and y . We have seen before that

$$\text{fl}(x + y) = (x + y)(1 + \delta) = x(1 + \delta) + y(1 + \delta) = x' + y'$$

Thus, the computed sum of two floating point numbers x and y is the exact sum of another two floating point numbers x' and y' . Because $|\delta| \leq \mu$, both x' and y' are close to x and y , respectively. Thus, we conclude that **the operation of adding two floating-point numbers is stable**. Similarly, floating-point subtraction, multiplication, and division are also backward stable.

Example 3.3.1 A Stable Algorithm: *Solution of an Upper Triangular System by Back-Substitution.*

The back-substitution method for solving an upper triangular system $Tx = b$ is backward stable. It can be shown that the computed solution \hat{x} satisfies

$$(T + E)\hat{x} = b,$$

where

$$|e_{ij}| \leq c\mu |t_{ij}|, \quad i, j = 1, \dots, n,$$

and c is a constant of order unity. Thus, the computed solution \hat{x} solves a nearby system. The back-substitution process is, therefore, backward stable.

Remark: The forward substitution method (**Algorithm 3.3.2**) for solving a lower triangular system has the same numerical stability property as above. **This algorithm is also stable.**

Example 3.3.2 An Unstable Algorithm: *Gaussian elimination without pivoting.*

It can be shown (see Section 3.5.2) that Gaussian elimination without pivoting applied to the linear system $Ax = b$ produces a solution \hat{x} such that

$$(A + E)\hat{x} = b$$

*with $\|E\|_\infty \leq cn^3\rho \|A\|_\infty \mu$. For an arbitrary matrix A , the number ρ above, called the **growth factor**, can be very large. When it happens, the computed solution \hat{x} does not solve a nearby problem.*

Example 3.3.3 An Unstable Algorithm: *Finding the eigenvalues of a matrix via its characteristic polynomial. The process is numerically unstable.*

There are two reasons: First, the characteristic polynomial of a matrix may not be obtained in a numerically stable way (see Chapter 4); second, the zeros of a polynomial can be extremely sensitive to small perturbations of the coefficients.

A well-known example of zero-sensitivity is the Wilkinson polynomial $P_n(x) = (x - 1)(x - 2) \cdots (x - 20)$. A small perturbation of 2^{-23} to the coefficient of x^{19} changes some of the zeros significantly: some of them even become complex. See Datta (1995, pp. 81-82) for details.

Remark: This example shows that the eigenvalues of a matrix should not be computed by finding the roots of its characteristic polynomial.

3.3.6 Conditioning of the Problem and Perturbation Analysis

From the preceding discussion we should not form the opinion that if a stable algorithm is used to solve a problem, then the computed solution will be accurate. As said before, a property of the problem called **conditioning** also contributes to the accuracy or inaccuracy of the computed result.

The conditioning of a problem is a property of the problem itself. It is concerned with how the solution of the problem will change if the input data contains some impurities. This concern arises from the fact that in practical applications, the data very often come from some experimental observations where the measurements can be subjected to disturbances (or “noises”) in the data. There are other sources of error also, such as **roundoff errors, discretization errors**, and so on. Thus, when a numerical analyst has a problem in hand to solve, he or she must frequently solve the problem not with the original data, but with data that have been perturbed. The question naturally arises: **What effects do these perturbations have on the solution?**

A theoretical study done by numerical analysts to investigate these effects, which is independent of the particular algorithm used to solve the problem, is called **perturbation analysis**. This study helps us detect whether a given problem is “bad” or “good” in the sense of whether small perturbations in the data will create a large or small change in the solution. Specifically we use the following standard definition.

Definition 3.3.4 *A problem (with respect to a given set of data) is called an **ill-conditioned** or **badly conditioned** problem if a small relative error in data causes a large relative error in the computed solution, regardless of the method of solution. Otherwise, it is called **well-conditioned**.*

Suppose a problem P is to be solved with an input c . Let $P(c)$ denote the value of the problem with the input c . Let δ_c denote the perturbation in c . Then P is said to be ill conditioned for

the input data c if the relative error $\frac{|P(c + \delta_c) - P(c)|}{|P(c)|}$ is much larger than the relative error in the data $\frac{|\delta_c|}{|c|}$

Note: The definition of conditioning is data-dependent. Thus, a problem that is ill conditioned for *one* set of data could be well-conditioned for *another* set.

3.3.7 Conditioning of the Problem, Stability of the Algorithm, and Accuracy of the Solution

As stated in the previous section, the conditioning of a problem is a property of the problem itself, and has nothing to do with the algorithm used to solve the problem. To a user, of course, the accuracy of the computed solution is of primary importance. However, the accuracy of a computed solution by a given algorithm is directly connected with both the stability of the algorithm and the conditioning of the problem. **If the problem is ill-conditioned, no matter how stable the algorithm is, the accuracy of the computed result cannot be guaranteed.** On the other hand, if a backward stable algorithm is applied to a well-conditioned problem, the computed result will be accurate.

Backward Stability and Accuracy

Stable Algorithm → Well-conditioned Problem ≡ Accurate Result.
 Stable Algorithm → Ill-conditioned Problem ≡ Possibly Inaccurate Result (Inaccuracy depends upon how ill-conditioned the problem is).

3.3.8 Conditioning of the Linear System and Eigenvalue Problems

The Condition Number of a Problem

Numerical analysts usually try to associate a number called the **condition number** with a problem. The condition number indicates whether the problem is ill- or well-conditioned. More specifically, the condition number gives a bound for the relative error in the solution when a small perturbation is applied to the input data.

We will now give results on the conditions of the linear system and eigenvalue problems.

Theorem 3.3.1 (General Perturbation Theorem)

Let ΔA and δb , be the perturbations, respectively, of the data A and b , and δx be the error in x . Assume that A is nonsingular and $\|\Delta A\| < \frac{1}{\|A^{-1}\|}$. Then

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\|A\| \|A^{-1}\|}{(1 - \|\Delta A\| \|A^{-1}\|)} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right).$$

Interpretation of the Theorem: The above Theorem says that if the relative perturbations in A and b are small, then the relative error in the solution x is dominated by the number $\|A\| \|A^{-1}\|$.

Definition 3.3.5 *The number $\|A\| \|A^{-1}\|$ is called the **condition number** of the linear system problem $Ax = b$ or just the condition number of A , and is denoted by $\text{Cond}(A)$.*

From Theorem above, it follows that **if $\text{Cond}(A)$ is large, then the system $Ax = b$ is ill-conditioned**, otherwise it is **well-conditioned**.

The condition number of a matrix certainly depends upon the norm of the matrix. However, roughly, if a matrix is ill-conditioned in one type of norm, it is ill-conditioned in other types as well. This is because the condition numbers in different norms are related. For example, for an $n \times n$ real matrix A , one can show that

$$\begin{aligned} \frac{1}{n} \text{Cond}_2(A) &\leq \text{Cond}_1(A) \leq n \text{Cond}_2(A), \\ \frac{1}{n} \text{Cond}_\infty(A) &\leq \text{Cond}_2(A) \leq n \text{Cond}_\infty(A), \\ \frac{1}{n^2} \text{Cond}_1(A) &\leq \text{Cond}_\infty(A) \leq n^2 \text{Cond}_1(A), \end{aligned}$$

where $\text{Cond}_p(A), p = 1, 2, \infty$ denotes the condition number in p -norm.

Next, we present the proof of the above theorem in the case $\Delta A = 0$. For the proof in the general case, see Datta (1995, pp. 249-250). We first restate the theorem in this special case.

Theorem 3.3.2 (Right Perturbation Theorem) *If δb and δx , are, respectively, the perturbations of b and x in the linear system $Ax = b$; and, A is assumed to be nonsingular and $b \neq 0$, then*

$$\frac{\|\delta x\|}{\|x\|} \leq \text{Cond}(A) \frac{\|\delta b\|}{\|b\|}.$$

Proof: We have

$$Ax = b,$$

and

$$A(x + \delta x) = b + \delta b.$$

The last equation can be written as

$$Ax + A\delta x = b + \delta b,$$

or

$$A\delta x = \delta b, \quad (\text{since } Ax = b).$$

That is,

$$\delta x = A^{-1} \delta b.$$

Taking a subordinate matrix-vector norm we get

$$\|\delta x\| \leq \|A^{-1}\| \|\delta b\|. \quad (3.3.1)$$

Again, taking the same norm on both sides of $Ax = b$, we get

$$\|Ax\| = \|b\|$$

or

$$\|b\| = \|Ax\| \leq \|A\| \|x\| \quad (3.3.2)$$

Combining (3.3.1) and (3.3.2) we have

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|\delta b\|}{\|b\|}. \quad (3.3.3)$$

Interpretation of Theorem 3.3.2

Theorem 3.3.2 says that a relative perturbation in the solution can be as large as $\text{Cond}(A)$ multiplied by the relative change in the vector b . Thus, if the condition number is not too large, then a small perturbation in the vector b will have very little effect on the solution. **On the other hand, if the condition number is large, then even a small perturbation in b might change the solution drastically.**

Example 3.3.4 An ill-conditioned linear system problem

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4.0001 & 2.002 \\ 1 & 2.002 & 2.004 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 8.0021 \\ 5.006 \end{pmatrix}$$

The exact solution $x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$. Change b to $b' = \begin{pmatrix} 4 \\ 8.0020 \\ 5.0061 \end{pmatrix}$.

Then the relative perturbation in b :

$$\frac{\|b' - b\|}{\|b\|} = \frac{\|\delta b\|}{\|b\|} = 1.379 \times 10^{-5} \quad (\text{small}).$$

If we solve the system $Ax' = b'$, we get

$$x' = x + \delta x = \begin{pmatrix} 3.0850 \\ -0.0436 \\ 1.0022 \end{pmatrix}.$$

(x' is completely different from x).

Note that the relative error in x : $\frac{\|\delta x\|}{\|x\|} = 1.3461$, (quite large!).

It is easily verified that the inequality in the above Theorem is satisfied:

$$\text{Cond}(A) \cdot \frac{\|\delta b\|}{\|b\|} = 4.4434, \text{Cond}(A) = 3.221 \times 10^5.$$

However, the predicted change is overly estimated.

Conditioning of Eigenvalues

Like the linear system problem, the eigenvalues and the eigenvectors of a matrix A can be ill-conditioned too.

The following result gives an overall sensitivity of the eigenvalues due to perturbations in the entries of A . For a proof, see Datta (1995) or Golub and Van Loan (1996).

Theorem 3.3.3 (Bauer-Fike). *Let $X^{-1}AX = D = \text{diag}(\lambda_1, \dots, \lambda_n)$. Then for any eigenvalue λ of $A + E \in \mathbb{C}^{n \times n}$, we have*

$$\min |\lambda_i - \lambda| \leq \text{Cond}_p(X) \|E\|,$$

where $\|\cdot\|_p$ is a p -norm.

The result says that the eigenvalues of A might be sensitive to small perturbations of the entries of A if the transforming matrix X is ill-conditioned.

Analysis of the conditioning of the individual eigenvalues and eigenvectors are rather involved. We just state here the conditioning of simple eigenvalues of a matrix.

Let λ_i be a **simple** eigenvalue of A . Then the condition number of λ_i , denoted by $\text{Cond}(\lambda_i)$, is defined to be: $\text{Cond}(\lambda_i) = \frac{1}{|y_i^T x_i|}$, where y_i and x_i are, respectively, the unit **left** and **right** eigenvectors associated with λ_i .

A well-known example of eigenvalue sensitivity is the **Wilkinson bidiagonal matrix**:

$$A = \begin{pmatrix} 20 & 20 & & & \\ & 19 & 20 & & 0 \\ 0 & \ddots & \ddots & \ddots & \\ & & \ddots & 20 & \\ & & & & 1 \end{pmatrix}$$

The eigenvalues of A are $1, 2, \dots, 20$.

A small perturbation E of the $(20, 1)$ th entry of A (say $E = 10^{-10}$) changes some of the eigenvalues drastically: **they even become complex** (see Datta (1995), pp. 84-85).

The above matrix A is named after the famous British numerical analyst **James H. Wilkinson**, who computed the condition numbers of the above eigenvalues, and found that some of the condition numbers were quite large; explaining the fact why they changed so much due to a small perturbation of just one entry of A .

Note: Though the eigenvalues of a nonsymmetric matrix can be ill-conditioned, the **eigenvalues of a symmetric matrix are well-conditioned**. (see Datta (1995), pp. 455-456).

3.4 LU Factorization

In this section, we describe a well-known matrix factorization, called the **LU factorization of a matrix** and in the next section, we will show how the LU factorization is used to solve an algebraic linear system.

3.4.1 LU Factorization using Gaussian Elimination

Let an $n \times n$ matrix A be factored into $LU : A = LU$, where L is a lower triangular matrix with 1's along the diagonal (unit lower triangular) and U is an $n \times n$ upper triangular matrix. Then the factorization is known as an **LU factorization of A** . A classical elimination technique, called **Gaussian elimination**, is used to achieve this factorization.

It can be shown (Golub and Van Loan (1996, pp. 97-98) that A has an LU factorization if the leading principal minors of A are nonsingular. Furthermore, if an LU factorization exists and A is nonsingular, then the LU factorization is unique.

Gaussian Elimination

There are $(n - 1)$ steps in the process.

Beginning with $A^{(0)} = A$, the matrices $A^{(1)}, \dots, A^{(n-1)}$ are constructed such that $A^{(k)}$ has zeros below the diagonal in the k th column. The final matrix $A^{(n-1)}$ will then be an upper triangular matrix U .

Denote $A^{(k)} = (a_{ij}^{(k)})$. The matrix $A^{(k)}$ is obtained from the previous matrix $A^{(k-1)}$ by multiplying the entries of the row k of $A^{(k-1)}$ with $m_{ik} = -\frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, i = k + 1, \dots, n$ and adding them to those of $(k + 1)$ through n . In other words,

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} + m_{ik} a_{kj}^{(k-1)}, i = k + 1, \dots, n; j = k + 1, \dots, n. \quad (3.4.1)$$

The entries m_{ik} are called **multipliers**. The entries $a_{kk}^{(k-1)}$ are called the **pivots**.

To see how an LU factorization, when it exists, can be obtained, we note (which is easy to see using the above relations) that

$$A^{(k)} = M_k A^{(k-1)}, \quad (3.4.2)$$

where M_k is a unit lower triangular matrix formed out of the multipliers. The matrix M_k is known as **elementary lower triangular matrix**. The matrix M_k can be written as

$$M_k = I + m_k e_k^T,$$

where e_k is the k -th unit vector, $e_i^T m_k = 0$ for $i \leq k$, and $m_k = (0, \dots, 0, m_{k+1,k}, \dots, m_{n,k})^T$.

Furthermore, $M_k^{-1} = I - m_k e_k^T$.

Using (3.4.2), we see that

$$U = A^{(n-1)} = M_{n-1} A^{(n-2)} = M_{n-1} M_{n-2} A^{(n-3)} \dots = M_{n-1} M_{n-2} \dots M_2 M_1 A.$$

Thus $A = (M_{n-1}M_{n-2} \cdots M_2M_1)^{-1}U = LU$, where $L = (M_{n-1}M_{n-2} \cdots M_2M_1)^{-1}$.

Since each of the matrices M_1 through M_{n-1} is a unit upper triangular matrix, so is L (**Note:** The product of two unit upper triangular matrix is an upper triangular matrix and the inverse of a unit upper triangular matrix is an upper triangular matrix).

Constructing L: The matrix L can be formed just from the multipliers, as shown below. **No explicit matrix inversion is needed.**

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ -m_{21} & 1 & 0 & \cdots & \cdots & 0 \\ -m_{31} & -m_{32} & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ -m_{n1} & -m_{n2} & -m_{n3} & \cdots & -m_{n,n-1} & 1 \end{pmatrix}$$

Difficulties with Gaussian Elimination without Pivoting

Gaussian elimination, as described above, fails if any of the pivots is zero, **it is worse yet if any pivot becomes close to zero:** In this case the method can be carried to completion, but the obtained results may be totally wrong.

Consider the following simple example: Let Gaussian elimination without pivoting be applied to $A = \begin{pmatrix} 0.0001 & 1 \\ 1 & 1 \end{pmatrix}$, using three decimal digit floating point arithmetic.

There is only one step. The multiplier $m_{21} = -1/10^{-4} = -10^4$. Let \hat{L} and \hat{U} be the computed versions of L and U . Then

$$\hat{U} = A^{(1)} = \begin{pmatrix} 0.0001 & 1 \\ 0 & 1 - 10^4 \end{pmatrix} = \begin{pmatrix} 0.0001 & 1 \\ 0 & -10^4 \end{pmatrix}.$$

(Note that $(1 - 10^4)$ gives -10^4 in three digit arithmetic). The matrix \hat{L} formed out the multiplier m_{21} is $\hat{L} = \begin{pmatrix} 1 & 0 \\ 10^4 & 1 \end{pmatrix}$.

The product of the computed \hat{L} and \hat{U} is: $\hat{L}\hat{U} = \begin{pmatrix} 0.0001 & 1 \\ 1 & 0 \end{pmatrix}$, which is different from A .

Note that the pivot $a_{11}^{(1)} = 0.0001$ is very close to zero (in three-digit arithmetic). This small pivot gave a large multiplier. This large multiplier, when used to update the entries of A , the number 1, which is much smaller compared to 10^4 , got wiped out in the subtraction of $1 - 10^4$ and the result was -10^4 .

Gaussian Elimination with Partial Pivoting

The above example suggests that disaster in Gaussian elimination without pivoting in the presence of a small pivot can perhaps be avoided by identifying a “**good pivot**” (a pivot as

large as possible) at each step, before the process of elimination is applied. The good pivot may be located among the entries in a column or among all the entries in a submatrix of the current matrix. In the former case, since the search is only partial, the method is called **partial pivoting**; in the latter case, the method is called **complete pivoting**. *It is important to note that the purpose of pivoting is to prevent large growth in the reduced matrices, which can wipe out the original data.* One way to do this is to keep multipliers less than 1 in magnitude, and this is exactly what is accomplished by pivoting.

We will discuss here only Gaussian elimination with partial pivoting, which also consists of $(n - 1)$ steps.

In fact, the process is just a slight modification of Gaussian elimination in the following sense: At each step, the largest entry (in magnitude) is identified among all the entries in the pivot column. This entry is then brought to the diagonal position of the current matrix by interchange of suitable rows and then using that entry as “pivot,” the elimination process is performed.

Thus if we set $A^{(0)} = A$, at step k ($k = 1, 2, \dots, n - 1$), first, the largest entry (in magnitude) $a_{r_k,k}^{(k-1)}$ is identified among all the entries of the column k (below the row $(k - 1)$) of the matrix $A^{(k-1)}$, this entry is then brought to the diagonal position by interchanging the rows k and r_k , and then the elimination process proceeds with $a_{r_k,k}^{(k-1)}$ as the pivot.

LU Factorization from Gaussian Elimination with Partial Pivoting

Since the interchange of two rows of a matrix is equivalent to premultiplying the matrix by a permutation matrix, the matrix $A^{(k)}$ is related to $A^{(k-1)}$ by the following relation:

$$A^{(k)} = M_k P_k A^{(k-1)}, k = 1, 2, \dots, n - 1,$$

where P_k is the permutation matrix obtained by interchanging the rows k and r_k of the identity matrix, and M_k is an elementary lower triangular matrix resulting from the elimination process. So,

$$\begin{aligned} U &= A^{(n-1)} = M_{n-1} P_{n-1} A^{(n-2)} = M_{n-1} P_{n-1} M_{n-2} P_{n-2} A^{(n-3)} \\ &= \dots = M_{n-1} P_{n-1} M_{n-2} P_{n-2} \dots M_2 P_2 M_1 P_1 A. \end{aligned}$$

Setting $M = M_{n-1} P_{n-1} M_{n-2} P_{n-2} \dots M_2 P_2 M_1 P_1$, we have the following factorization of A :

$$U = MA$$

The above factorization can be written in the form: $PA = LU$, where $P = P_{n-1} P_{n-2} \dots P_2 P_1$, $U = A^{(n-1)}$, and the matrix L is a unit lower triangular matrix formed out of the multipliers. For details, see Golub and Van Loan (1996,).

For $n = 4$, the reduction of A to the upper triangular matrix U can be schematically described as follows:

$$\begin{aligned}
1. \quad A &\xrightarrow{P_1} P_1 A \xrightarrow{M_1} M_1 P_1 A = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{pmatrix} = A^{(1)} \\
2. \quad A^{(1)} &\xrightarrow{P_2} P_2 A^{(1)} \xrightarrow{M_2} M_2 P_2 A^{(1)} = M_2 P_2 M_1 P_1 A = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix} = A^{(2)} \\
3. \quad A^{(2)} &\xrightarrow{P_3} P_3 A^{(2)} \xrightarrow{M_3} M_3 P_3 A^{(2)} = M_3 P_3 M_2 P_2 M_1 P_1 A = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix} = A^{(3)} = U.
\end{aligned}$$

The only difference between L here and the matrix L from Gaussian elimination without pivoting is that the multipliers in the k th column are now permuted according to the permutation matrix $\tilde{P}_k = P_{n-1} P_{n-2} \cdots P_{k+1}$.

Thus to construct L , again no explicit products or matrix inversions are needed. We illustrate this below.

Consider the case $n = 4$, and suppose P_2 interchange rows 2 and 3, and P_3 interchanges rows 3 and 4.

$$\text{The matrix } L \text{ is then given by } L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -m_{31} & 1 & 0 & 0 \\ -m_{21} & -m_{42} & 1 & 0 \\ -m_{41} & -m_{32} & -m_{34} & 1 \end{pmatrix}.$$

Example 3.4.1

$$A = \begin{pmatrix} 1 & 2 & 4 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

$k = 1$.

1. The pivot entry is 7: $r_1 = 3$.

$$\begin{aligned}
2. \quad \text{Interchange rows 3 and 1. } P_1 &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, P_1 A = \begin{pmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 4 \end{pmatrix} \\
3. \quad \text{Form the multipliers: } a_{21} &\equiv m_{21} = -\frac{4}{7}, \quad a_{31} \equiv m_{31} = -\frac{1}{7}.
\end{aligned}$$

$$4. \quad A^{(1)} = M_1 P_1 A = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{4}{7} & 1 & 0 \\ -\frac{1}{7} & 0 & 1 \end{pmatrix} \begin{pmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 4 \end{pmatrix} \equiv \begin{pmatrix} 7 & 8 & 9 \\ 0 & \frac{3}{7} & \frac{6}{7} \\ 0 & \frac{6}{7} & \frac{19}{7} \end{pmatrix}.$$

$k = 2$.

1. The pivot entry is $\frac{6}{7}$, $r_2 = 3$.

2. Interchange rows 2 and 3. $P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$, $P_2 A^{(1)} = \begin{pmatrix} 7 & 8 & 9 \\ 0 & \frac{6}{7} & \frac{19}{7} \\ 0 & \frac{3}{7} & \frac{6}{7} \end{pmatrix}$.

3. Form the multiplier: $m_{32} = -\frac{1}{2}$

$$A^{(2)} = M_2 P_2 A^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 7 & 8 & 9 \\ 0 & \frac{6}{7} & \frac{19}{7} \\ 0 & \frac{3}{7} & \frac{6}{7} \end{pmatrix} = \begin{pmatrix} 7 & 8 & 9 \\ 0 & \frac{6}{7} & \frac{19}{7} \\ 0 & 0 & -\frac{1}{2} \end{pmatrix}.$$

$$\text{Form } L = \begin{pmatrix} 1 & 0 & 0 \\ -m_{31} & 1 & 0 \\ -m_{21} & -m_{32} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{7} & 1 & 0 \\ \frac{4}{7} & \frac{1}{2} & 1 \end{pmatrix}.$$

$$P = P_2 P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Verify. $PA = \begin{pmatrix} 7 & 8 & 9 \\ 1 & 2 & 4 \\ 4 & 5 & 6 \end{pmatrix} = LU$.

Flop-Count. Gaussian elimination with partial pivoting requires only $\frac{2}{3}n^3$ flops. Furthermore, the process with partial pivoting requires at most $O(n^2)$ comparisons for identifying the pivots.

Stability of Gaussian Elimination.

The stability of Gaussian elimination algorithms is better understood by measuring the growth of the elements in the reduced matrices $A^{(k)}$. (Note that although pivoting keeps the multipliers bounded by unity, the elements in the reduced matrices still can grow arbitrarily).

Definition 3.4.1 The growth factor ρ is the ratio of the largest element (in magnitude) of $A, A^{(1)}, \dots, A^{(n-1)}$ to the largest element (in magnitude) of A : $\rho = \frac{\max(\alpha, \alpha_1, \alpha_2, \dots, \alpha_{n-1})}{\alpha}$, where $\alpha = \max_{i,j} |a_{ij}|$, and $\alpha_k = \max_{i,j} |a_{ij}^{(k)}|$.

The growth factor ρ can be arbitrarily large for Gaussian elimination without pivoting.

Note that ρ for the matrix $A = \begin{pmatrix} 0.0001 & 1 \\ 1 & 1 \end{pmatrix}$ without pivoting is 10^4 .

Thus, Gaussian elimination without pivoting is, in general, unstable.

Note: Though Gaussian elimination without pivoting is unstable for arbitrary matrices, there are two classes of matrices, the **diagonally dominant matrices** and the **symmetric positive**

definite matrices, for which the process can be shown to be stable. The growth factor of a diagonally dominant matrix is bounded by 2 and that the growth factor of a symmetric positive definite matrix is 1.

The next question is: How large the growth factor can be for Gaussian elimination with partial pivoting?

The growth factor ρ for Gaussian elimination with partial pivoting can be as large as 2^{n-1} : $\rho \leq 2^{n-1}$.

Though matrices for which this bound is attained can be constructed (see Datta (1995)), such matrices are rare in practice. Indeed, in many practical examples, the elements of the matrices $A^{(k)}$ very often continue to decrease in size. **Thus, Gaussian elimination with partial pivoting is not unconditionally stable in theory; in practice, however, it can be considered as a stable algorithm.**

MATLAB Note: The MATLAB command $[L, U, P] = \text{lu}(A)$ returns lower triangular matrix L , upper triangular matrix U , and permutation matrix P such that $PA = LU$.

3.4.2 The Cholesky Factorization

Every symmetric positive definite matrix A can be factored into

$$A = HH^T$$

where H is a lower triangular matrix with positive diagonal entries.

This factorization of A is known as the **Cholesky factorization**. Since, the growth factor for Gaussian elimination of a symmetric positive definite matrix is 1, **Gaussian elimination can be safely used to compute the Cholesky factorization of a symmetric positive definite matrix**. Unfortunately, no advantage of symmetry of the matrix A can be taken in the process.

In practice, the entries of the lower triangular matrix H , called the **Cholesky factor**, are computed directly from the relation $A = HH^T$. The matrix H is computed row by row. The algorithm is known as the Cholesky algorithm. See Datta (1995), pp. 222-223 for details.

Algorithm 3.4.1 *The Cholesky Algorithm*

Input: A —A symmetric positive definite matrix

Output: H —The Cholesky factor

For $k = 1, 2, \dots, n$ do

 For $i = 1, 2, \dots, k-1$ do

$$h_{ki} = \frac{1}{h_{ii}}(a_{ki} - \sum_{j=1}^{i-1} h_{ij}h_{kj})$$

$$h_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} h_{kj}^2}$$

End
End

Flop-count and Numerical Stability. Algorithm 3.4.1 requires only $\frac{n^3}{3}$ flops. The algorithm is numerically stable.

MATLAB and MATCOM Notes: Algorithm 3.4.1 has been implemented in MATCOM program **choles**. MATLAB function **chol** also can be used to compute the Cholesky factor. However, note that $L = \text{chol}(A)$ computes an upper triangular matrix R such that $A = R^T R$.

3.4.3 LU Factorization of an Upper Hessenberg Matrix

Recall that $H = (h_{ij})$ is an upper Hessenberg matrix if $h_{ij} = 0$ whenever $i > j + 1$. Thus Gaussian elimination scheme applied to an $n \times n$ upper Hessenberg matrix requires zeroing of only the nonzero entries on the subdiagonal. This means at each step, after a possible interchange of rows, just a multiple of the row containing the pivot has to be added to the next row.

Specifically, Gaussian elimination scheme with partial pivoting for an $n \times n$ upper Hessenberg matrix $H = (h_{ij})$ is as follows:

Algorithm 3.4.2 LU Factorization of an upper Hessenberg matrix

Input: H —An $n \times n$ upper Hessenberg matrix

Output: U —The upper triangular matrix U of LU factorization of H , stored over the upper part of H . The subdiagonal entries of H contain the multipliers.

For $k = 1, 2, \dots, n - 1$ do

1. Interchange $h_{k,j}$ and $h_{k+1,j}$, if $|h_{k,k}| < |h_{k+1,k}|$, $j = k, \dots, n$.
2. Compute the multiplier and store it over $h_{k+1,k}$: $h_{k+1,k} \equiv -\frac{h_{k+1,k}}{h_{k,k}}$.
3. Update $h_{k+1,j}$: $h_{k+1,j} \equiv h_{k+1,j} + h_{k+1,k} \cdot h_{k,j}$, $j = k + 1, \dots, n$.

End.

Example 3.4.2 $H = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 0 & 2 & 2 \end{pmatrix}$.

$k = 1$. Since $h_{11} = 1 < h_{21} = 3$, the first and second rows are interchanged.

$$H = \begin{pmatrix} 3 & 4 & 5 \\ 1 & 2 & 3 \\ 0 & 2 & 2 \end{pmatrix}.$$

The multiplier $h_{21} \equiv -\frac{h_{21}}{h_{11}} = -\frac{1}{3}$.

Update: $h_{22} \equiv \frac{2}{3}, h_{23} = \frac{4}{3}$.

$$H \text{ (updated)} = \begin{pmatrix} 3 & 4 & 5 \\ -\frac{1}{3} & \frac{2}{3} & \frac{4}{3} \\ 0 & 2 & 2 \end{pmatrix}$$

$k = 2$.

1. Since $h_{22} = \frac{2}{3} < h_{33} = 2$, interchange h_{22} with h_{32} and h_{23} with h_{33} .

$$H \text{ (updated)} = \begin{pmatrix} 3 & 4 & 5 \\ -\frac{1}{3} & 2 & 2 \\ 0 & \frac{2}{3} & \frac{4}{3} \end{pmatrix}$$

2. The multiplier $h_{32} = -\frac{h_{32}}{h_{22}} = -\frac{1}{3}$.

3. Update $h_{33} = \frac{2}{3}$.

$$H \text{ (updated)} = \begin{pmatrix} 3 & 4 & 5 \\ -\frac{1}{3} & 2 & 2 \\ 0 & -\frac{1}{3} & \frac{2}{3} \end{pmatrix}.$$

$$U = \begin{pmatrix} 3 & 4 & 5 \\ 0 & 2 & 2 \\ 0 & 0 & \frac{2}{3} \end{pmatrix}, L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{3} & 1 \end{pmatrix}$$

$$P = P_2 P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

$$\text{Verify: } PH = \begin{pmatrix} 3 & 4 & 5 \\ 0 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix} = LU.$$

Flop-count and Stability. The above algorithm requires n^2 flops.

It can be shown (Higham (1996), pp. 182, Wilkinson (1965), pp. 218)), that the growth factor ρ of a Hessenberg matrix for Gaussian elimination with partial pivoting is less than or equal to n . Thus, computing LU factorization of a Hessenberg matrix using Gaussian elimination with partial pivoting is an efficient and a numerically stable procedure.

3.5 Numerical Solution of the Linear System $Ax = b$.

Given an $n \times n$ matrix A and the n -vector b , the algebraic linear system problem is the problem of finding an n -vector x such that $Ax = b$.

The principal uses of the LU factorization of a matrix A are: **solving the algebraic linear system $Ax = b$** , **finding the determinant of a matrix**, and **finding the inverse of A** .

We will discuss first how $Ax = b$ can be solved using the LU factorization of A .

The following theorem gives results on the existence and uniqueness of the solution x of $Ax = b$. Proof can be found in any linear algebra text.

Theorem 3.5.1 (Existence and Uniqueness Theorem).

The system $Ax = b$ has a solution if and only if $\text{rank}(A) = \text{rank}(A, b)$. The solution is unique if and only if A is invertible.

3.5.1 Solving $Ax = b$ using the inverse of A

The above theorem suggests that the unique solution x of $Ax = b$ be computed as $x = A^{-1}b$. Unfortunately, **computationally this is not a practical idea. It generally involves more computations and gives less accurate answers.**

This can be illustrated by the following trivial example:

Consider solving $3x = 27$.

The exact answer is: $x = \frac{27}{3} = 9$. Only one flop (one division) is needed in this process. On the other hand, if the problem is solved by writing it in terms of the inverse of A , we then have $x = \frac{1}{3} \times 27 = 0.3333 \times 27 = 8.9991$ (in four digit arithmetic), a less accurate answer. Moreover, the process will need two flops: one division and one multiplication.

3.5.2 Solving $Ax = b$ using Gaussian Elimination with Partial Pivoting

Since Gaussian elimination without pivoting does not always work and, even when it works, might give an unacceptable answer in certain instances, we only discuss solving $Ax = b$ using Gaussian elimination with partial pivoting.

We have just seen that Gaussian elimination with partial pivoting, when used to triangularize A , yields a factorization $PA = LU$. In this case, the system $Ax = b$ is equivalent to the two triangular systems:

$$Ly = Pb = b' \text{ and } Ux = y.$$

Thus, to solve $Ax = b$ using Gaussian elimination with partial pivoting the following two steps need to be performed in the sequence.

Step 1. Find the factorization $PA = LU$ using Gaussian eliminating with partial pivoting.

Step 2. Solve the lower triangular system: $Ly = Pb = b'$ first, followed by the upper triangular system: $Ux = y$.

Forming the vector b' :

The vector b' is just the permuted version of b . So, to obtain b' , all that needs to be done is to permute the entries of b in the same way as the rows of the matrices $A^{(k)}$ have been interchanged. This is illustrated in the following example.

Example 3.5.1 Solve the following system using Gaussian elimination with partial pivoting:

$$\begin{aligned}x_1 + 2x_2 + 4x_3 &= 7 \\4x_1 + 5x_2 + 6x_3 &= 15 \\7x_1 + 8x_2 + 9x_3 &= 24\end{aligned}$$

Here

$$A = \begin{pmatrix} 1 & 2 & 4 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, b = \begin{pmatrix} 7 \\ 15 \\ 24 \end{pmatrix}$$

Using the results of Example 3.4.1, we have

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{7} & 1 & 0 \\ \frac{4}{7} & \frac{1}{2} & 1 \end{pmatrix}, U = \begin{pmatrix} 7 & 8 & 9 \\ 0 & \frac{6}{7} & \frac{19}{7} \\ 0 & \frac{6}{7} & \frac{19}{7} \\ 0 & 0 & -\frac{1}{2} \end{pmatrix}$$

$$\text{Since } r_1 = 3, \text{ and } r_2 = 3, b' = \begin{pmatrix} 24 \\ 7 \\ 15 \end{pmatrix}.$$

Note that to obtain b' , first the 1st and 3rd components of b were permuted, according to $r_1 = 3$ (which means the interchange of rows 1 and 3), followed by the permutation of the components 2 and 3, according to $r_2 = 3$ (which means the interchange of the rows 2 and 3).

$$Ly = b' \text{ gives } y = \begin{pmatrix} 24 \\ 3.5714 \\ -0.5000 \end{pmatrix}, \text{ and } Ux = y \text{ gives } x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Flop Count. The factorization process requires about $\frac{2}{3}n^3$ flops. The solution of each of the triangular systems $Ly = b'$ and $Ux = y$ requires n^2 flops. Thus, the solution of the linear system $Ax = b$ using Gaussian elimination with partial pivoting requires about $2n^3/3 + O(n^2)$ flops. Also, the process requires $O(n^2)$ comparisons for pivot-identifications.

Stability of Gaussian Elimination Scheme for $Ax = b$

We have seen that the growth factor ρ determines the stability of the triangularization procedure. The next question is, how does the growth factor affect the solution procedure of $Ax = b$ using such a triangularization?

The answer is given in the following:

The computed solution \hat{x} of the linear system $Ax = b$, using Gaussian elimination, satisfies $(A + E)\hat{x} = b$ where $\|E\|_\infty \leq c(n^3 + 5n^2)\rho \|A\|_\infty \mu$, and c is a small constant (see Datta (1995), Chapter 11). Here ρ is the growth factor and μ is the machine precision.

Since the large growth factor ρ for Gaussian elimination with partial pivoting is rare in practice, **for all practical purposes, Gaussian elimination with partial pivoting for the linear system $Ax = b$ is a numerically stable procedure.**

3.5.3 Solving a Hessenberg Linear System

Certain control computations such as **computing the frequency response of a matrix** (see Chapter 5) require solution of a Hessenberg linear algebraic system. We have just seen that the LU factorization of a Hessenberg matrix requires only $O(n^2)$ flops and Gaussian elimination with partial pivoting is safe; because, the growth factor in this case is at most 2. Thus, **a Hessenberg system can be solved using Gaussian elimination with partial pivoting using $O(n^2)$ flops and in a numerically stable way.**

3.5.4 Solving $AX = B$.

In many practical situations, one faces the problem of solving multiple linear systems: $AX = B$. Here A is $n \times n$ and nonsingular and B is $n \times p$. Since each of the systems here has the same coefficient matrix A , to solve $AX = B$, we need to factor A just once. The following scheme, then, can be used.

Partition $B = (b_1, \dots, b_p)$.

Step 1. Factorize A using Gaussian elimination with partial pivoting: $PA = LU$

Step 2. For $k = 1, \dots, p$ do

Solve $Ly = Pb_k$

Solve $Ux_k = y$

End

Step 3. Form $X = (x_1, \dots, x_p)$.

3.5.5 Finding the Inverse of A

The inverse of an $n \times n$ nonsingular matrix A can be obtained as a special case of the above method. Just set $B = I_{n \times n}$. Then $X = A^{-1}$.

3.5.6 Computing the Determinant of A

The determinant of matrix A can be immediately computed, once the LU factorization of A is available. Thus, if Gaussian elimination with partial pivoting is used giving $PA = LU$, then $\det(A) = (-1)^r \prod_{i=1}^n u_{ii}$, where r is the number of row interchanges in the partial pivoting process.

3.5.7 Iterative Refinement

Once the system $Ax = b$ is solved using Gaussian elimination, it is suggested that the computed solution be refined iteratively to a desired accuracy using the following procedure. The procedure is fairly inexpensive and requires only $O(n^2)$ flops for each iteration.

Let x be the computed solution of $Ax = b$ obtained by using Gaussian elimination with partial pivoting factorization: $PA = LU$.

For $k = 1, 2, \dots$, do until desired accuracy.

1. Compute the residual $r = b - Ax$ (in double precision).
2. Solve $Ly = Pr$ for y .
3. Solve $Uz = y$ for z .
4. Update the solution $x \equiv x + z$.

3.6 The QR Factorization

Recall that a square matrix O is said to be an **orthogonal matrix** if $OO^T = O^TO = I$.

Given an $m \times n$ matrix A there exist an $m \times m$ orthogonal matrix Q and an $m \times n$ upper triangular matrix R such that $A = QR$.

Such a factorization of A is called the **QR factorization**.

If $m \geq n$, and if the matrix Q is partitioned as $Q = [Q_1, Q_2]$, where Q_1 is the matrix of the first n columns of Q , and if R_1 is defined by $R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$, where R_1 is $n \times n$ upper triangular, then $A = Q_1R_1$.

This QR factorization is called the “**economy size**” or the “**thin**” QR factorization of A .

The following theorem gives condition for uniqueness of the “thin” QR factorization. For a proof of the Theorem, see Golub and Van Loan (1996).

Theorem 3.6.1 *Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$ have full rank. Then the thin QR factorization*

$$A = Q_1R_1$$

is unique. Furthermore, the diagonal entries of R_1 are all positive.

There are several ways to compute the QR factorization of a matrix. Householder’s and Givens’ methods can be used to compute both types of QR factorizations. On the other hand, the classical Gram-Schmidt (CGS) and the modified Gram-Schmidt (MGS) compute $Q \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{R}^{n \times n}$ such that $A = QR$.

The MGS has better numerical properties than CGS. We will not discuss them here. The readers are referred to the book (Datta (1995), pp. 339-343). We will discuss Householder’s and Givens’ methods in the sequel.

3.6.1 Householder Matrices

Definition 3.6.1 A matrix of the form $H = I - \frac{2uu^T}{u^Tu}$,

where u is a nonzero vector, is called a **Householder matrix**, after the celebrated American numerical analyst Alston Householder.

A Householder matrix is also known as an **Elementary Reflector** or a **Householder transformation**.

It is easy to see that a Householder matrix H is **symmetric** and **orthogonal**.

A Householder matrix H is an important tool to create zeros in a vector:

Given $x = (x_1, x_2, \dots, x_n)^T$, the Householder matrix $H = I - 2\frac{uu^T}{u^Tu}$,

where

$u = x + \text{sign}(x_1) \|x\|_2 e_1$ is such that

$$Hx = (\sigma, 0, \dots, 0)^T, \text{ where } \sigma = -\text{sgn}(x_1) \|x\|_2.$$

Schematically, $x \xrightarrow{H} Hx = (\|x\|_2, 0, \dots, 0)^T$.

Forming Matrix-Vector and Matrix-Matrix Products With a Householder Matrix

A remarkable computational advantage involving Householder matrices is that neither a matrix-vector product with a Householder matrix H nor the matrix product HA (or AH) needs to be explicitly formed, as can be seen from the followings:

1. $Hx = \left(I - 2\frac{uu^T}{u^Tu}\right)x = x - \beta u(u^Tx)$, where $\beta = \frac{2}{u^Tu}$.
2. $HA = (I - \beta uu^T)A = A - \beta uu^TA = A - \beta uv^T$, where $v = A^Tu$.
3. $AH = A(I - \beta uu^T) = A - \beta vu^T$, where $v = Au$.

From above, we immediately see that the matrix product HA or AH requires only $O(n^2)$ flops, a substantial saving compared to $2n^3$ flops that are required to compute the product of two arbitrary matrices.

3.6.2 The Householder QR Factorization

Householder's method for the QR factorization of matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, consists of constructing Householder matrices H_1, H_2, \dots, H_n successively such that

$$H_n H_{n-1} \cdots H_1 A = R$$

is an $m \times n$ upper triangular matrix. If $H_1 H_2 \cdots H_n = Q$, then Q is an orthogonal matrix (since each H_i is orthogonal) and from above, we have $Q^T A = R$ or $A = QR$. Note that $R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$, where $R_1 \in \mathbb{R}^{n \times n}$ and is upper triangular.

We now sketch a procedure of forming the matrices H_1 through H_n to achieve the QR factorization of A .

First, partition the matrix as $A = (a_1, A'_1)$, where a_1 is the 1st column of A and find a Householder matrix H_1 such that $H_1 a_1 = \begin{pmatrix} * \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.

$$\text{holder matrix } H_1 \text{ such that } H_1 a_1 = \begin{pmatrix} * \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

$$\text{Then } H_1 A = (H_1 a_1, H_1 A'_1) = \begin{pmatrix} * & * \\ \hline 0 & \\ \vdots & A_2 \\ 0 & \end{pmatrix} = A^{(1)}.$$

Next, partition A_2 as $A_2 = (\hat{a}_2, A'_2)$, where \hat{a}_2 is the 1st column of A_2 , and find a Householder

$$\text{matrix } \hat{H}_2 \text{ such that } \hat{H}_2 \hat{a}_2 = \begin{pmatrix} * \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \text{ If } H_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \hat{H}_2 & \\ 0 & & & \end{pmatrix}, \text{ then } A^{(1)} H_2 = H_2 A^{(1)} =$$

$$H_2 H_1 A = \begin{pmatrix} * & & * \\ \vdots & * & \\ \vdots & \vdots & \boxed{A_3} \\ 0 & 0 & \end{pmatrix} = A^{(2)}.$$

The process is fairly general and can be continued until the QR factorization of A is obtained. For $n = 4, m = 2$, schematically, reduction to the triangular form is described as follows:

$$A \xrightarrow{H_1} H_1 A = \begin{pmatrix} * & * \\ 0 & * \\ 0 & * \\ 0 & * \end{pmatrix} = A^{(1)} = \begin{pmatrix} * & * \\ \hline 0 & \\ 0 & A_2 \\ 0 & \end{pmatrix}.$$

$$A^{(1)} \xrightarrow{H_2} H_2 A^{(1)} = H_2 H_1 A = \begin{pmatrix} * & * \\ 0 & * \\ 0 & 0 \\ 0 & 0 \end{pmatrix} = A^{(2)} = R.$$

Note: To compute H_2 , first \hat{H}_2 is computed acting on the vector $\hat{a}_2 = \begin{pmatrix} * \\ * \\ * \end{pmatrix}$, then H_2 is formed as $H_2 = \text{diag}(1, \hat{H}_2)$, as shown above.

Flop-Count: The Householder QR factorization method requires approximately $2n^2(m - \frac{n}{3})$ flops just to compute the triangular matrix R .

Note: The above procedure does not give the orthogonal matrix Q explicitly. The matrix Q can be computed, if required, as $Q = H_1 \cdots H_n$ by forming the product implicitly, as shown in

Section 3.6.1.

It should be noted that in a majority of practical applications, it is sufficient to have Q in this factored form; in many applications, Q is not needed at all. If Q is needed explicitly, another $4(m^2n - mn^2 + \frac{n^3}{3})$ flops will be required.

Numerical Stability: The Householder QR factorization method computes the QR factorization of a slightly perturbed matrix. Specifically, it can be shown (Wilkinson (1965) pp. 236) that, if \hat{R} denotes the computed R , then there exists an orthogonal \hat{Q} such that

$$A + E = \hat{Q}\hat{R},$$

where

$$\|E\|_2 \approx \mu \|A\|_2.$$

The algorithm is thus stable.

MATLAB Notes: $[Q, R] = qr(A)$ computes the QR factorization of A , using Householder's method.

Example 3.6.1 $A = \begin{pmatrix} 1 & 1 \\ 0.0001 & 0 \\ 0 & 0.0001 \end{pmatrix}$

$k = 1$

Form H_1 :

$$u_1 = \begin{pmatrix} 1 \\ 0.0001 \\ 0 \end{pmatrix} + \sqrt{1 + (0.0001)^2} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0.0001 \\ 0 \end{pmatrix}$$

$$\text{Update } A \equiv A^{(1)} = H_1 A = \left(I - \frac{2u_1 u_1^T}{u_1^T u_1}\right) A = \begin{pmatrix} -1 & -1 \\ 0 & -0.0001 \\ 0 & 0.0001 \end{pmatrix}$$

$k = 2$

Form H_2 :

$$u_2 = \begin{pmatrix} -0.0001 \\ 0.0001 \end{pmatrix} - \sqrt{(-0.0001)^2 + (0.0001)^2} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 10^{-4} \begin{pmatrix} -2.4141 \\ .1000 \end{pmatrix}$$

$$\hat{H}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - 2 \frac{u_2 u_2^T}{u_2^T u_2} = \begin{pmatrix} -0.7071 & 0.7071 \\ 0.7071 & 0.7071 \end{pmatrix}, H_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.7071 & 0.7071 \\ 0 & 0.7071 & 0.7071 \end{pmatrix}$$

$$\text{Update } A \equiv A^{(2)} = H_2 A^{(1)} = \begin{pmatrix} -1 & -1 \\ 0 & 0.0001 \\ 0 & 0 \end{pmatrix}$$

Form Q and R :

$$Q = H_1 H_2 = \begin{pmatrix} -1 & 0.0001 & -0.0001 \\ -0.0001 & -0.7071 & 0.7071 \\ 0 & 0.7071 & 0.7071 \end{pmatrix}$$

$$R = \begin{pmatrix} -1 & -1 \\ 0 & 0.0001 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \text{ where } R_1 = \begin{pmatrix} -1 & -1 \\ 0 & 0.0001 \end{pmatrix}$$

Complex QR Factorization

If $x \in \mathbb{C}^n$, and $x_1 = re^{i\theta}$, then it is easy to see that the Householder matrix $H = I - \beta vv^*$, where $v = x \pm e^{i\theta} \|x\|_2 e_1$ and $\beta = \frac{2}{v^*v}$, is such that $Hx = \mp ve^{i\theta} \|x\|_2 e_1$.

Using the above formula, the Householder QR factorization method for a real matrix A , described in the last section, can be easily adapted to a complex matrix. The details are left to the readers.

The process of complex QR factorization of an $m \times n$ matrix, $m \geq n$, using Householder's method requires $8n^2(m - \frac{n}{3})$ real flops.

3.6.3 Givens Matrices

Definition 3.6.2 A matrix of the form

$$J(i, j, c, s) = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & & & & \cdots & \vdots & \\ \vdots & \vdots & & & & & \cdots & \vdots & \\ 0 & 0 & 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & \vdots & \vdots & & & & \cdots & \vdots & \\ 0 & 0 & 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & \vdots & \vdots & & & & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 & \cdots & 1 \end{pmatrix} \leftarrow \begin{array}{l} i^{th} \\ j^{th} \end{array}$$

where $c^2 + s^2 = 1$, is called a **Givens matrix**, after the name of the numerical analyst Wallace Givens.

Since one can choose $c = \cos \theta$ and $s = \sin \theta$ for some θ , the above Givens matrix can be conveniently denoted by $J(i, j, \theta)$. Geometrically, the matrix $J(i, j, \theta)$ rotates a pair of coordinate

axes (i^{th} unit vector as its x -axis and the j^{th} unit vector as its y -axis) through the given angle θ in the (i, j) plane. That is why, the Givens matrix $J(i, j, \theta)$ is commonly known as a **Givens Rotation or Plane Rotation in the (i, j) plane**.

Thus, when an n -vector $x = (x_1, x_2, \dots, x_n)^T$ is premultiplied by the Givens rotation $J(i, j, \theta)$, only the i^{th} and j^{th} components of x are affected; the other components remain unchanged.

Also, note that since $c^2 + s^2 = 1$; $J(i, j, \theta) \cdot J(i, j, \theta)^T = I$. **So, the Givens matrix $J(i, j, \theta)$ is orthogonal.**

Zeroing the Entries of a 2×2 vector using a Givens Matrix

If $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ is a vector, then it is a matter of simple verification that, with

$$c = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}, \quad \text{and } s = \frac{x_2}{\sqrt{x_1^2 + x_2^2}},$$

the Givens rotation $J(1, 2, \theta) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$ is such that $J(1, 2, \theta)x = \begin{pmatrix} * \\ 0 \end{pmatrix}$.

The preceding formula for computing c and s might cause some **underflow** or **overflow**. However, the following simple rearrangement of the formula might prevent that possibility.

1. If $|x_2| \geq |x_1|$, compute $t = \frac{x_1}{x_2}$, $s = \frac{1}{\sqrt{1+t^2}}$, and take $c = st$.
2. If $|x_2| < |x_1|$, compute $t = \frac{x_2}{x_1}$, $c = \frac{1}{\sqrt{1+t^2}}$, and take $s = ct$.

Implicit Construction of JA.

If A is $\mathbb{R}^{m \times n}$ and $J(i, j, c, s) \in \mathbb{R}^{m \times m}$, then the update $A \equiv J(i, j, c, s)A$ can be computed implicitly as follows:

For $k = 1, \dots, n$ do

$$\begin{aligned} a &\equiv a_{ik} \\ b &\equiv a_{jk} \\ a_{ik} &\equiv ac + bs \\ a_{jk} &\equiv -as + bc \end{aligned}$$

End

MATCOM Note: The above computation has been implemented in MATCOM program PGIVMUL.

3.6.4 The QR Factorization Using Givens Rotations

Assume that $A \in \mathbb{R}^{m \times n}$, $m \geq n$. The basic idea is just like Householder's: Compute orthogonal matrices Q_1, Q_2, \dots, Q_n , using Givens rotations such that $A^{(1)} = Q_1 A$ has zeros below the $(1, 1)$ entry in the first column, $A^{(2)} = Q_2 A^{(1)}$ has zeros below the $(2, 2)$ entry in the second column, and so on. Each Q_i is generated as a product of Givens rotations. One way to form $\{Q_i\}$ is:

$$\begin{aligned} Q_1 &= J(1, m, \theta) J(1, m-1, \theta) \cdots J(1, 2, \theta) \\ Q_2 &= J(2, m, \theta) J(2, m-1, \theta) \cdots J(2, 3, \theta) \end{aligned}$$

and so on.

Then

$$\begin{aligned} R &= A^{(n)} = Q_n A^{(n-1)} = Q_n Q_{n-1} A^{(n-2)} = \dots \\ &= Q_n Q_{n-1} \cdots Q_2 Q_1 A = Q^T A, \text{ where } Q = Q_1^T Q_2^T \cdots Q_n^T. \end{aligned}$$

Algorithm 3.6.1 *Givens QR Factorization*

Input: A -An $m \times n$ matrix

Outputs: R -An $m \times n$ upper triangular matrix stored over A .

Q -An $m \times m$ orthogonal matrix in factored form defined by the Givens parameters c, s , and the indices k , and l .

Step 1. For $k = 1, 2, \dots, n$ do

For $l = k+1, \dots, m$ do

Step 1.1. Find c and s using the formulas given in **Section 3.6.3**

so that

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a_{kk} \\ a_{lk} \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix}$$

Step 1.2 Save the indices k and l and the numbers c and s

Step 1.3 Update A using the implicit construction as shown above:

$$A \equiv J(i, j, c, s) A$$

End

End

Step 2. Set $R \equiv A$.

Forming the Matrix Q

If the orthogonal matrix Q is needed explicitly, then it can be computed from the product $Q = Q_1^T Q_2^T \cdots Q_n^T$, where each Q_i is the product of $m-i$ Givens rotations: $Q_i = J(i, m, \theta) J(i, m-1, \theta) \cdots J(i, i+1, \theta)$

Flop-count. The algorithm requires $3n^2(m-n/3)$ flops; $m \geq n$. This count, of course, does not include computation of Q .

Numerical Stability. The algorithm is stable. It can be shown (Wilkinson (1965), pp. 240) that for $m = n$ the computed \hat{Q} and \hat{R} satisfy $\hat{R} = \hat{Q}^T(A + E)$, where $\|E\|_F$ is small.

MATCOM Note: The above algorithm has been implemented in MATCOM program GIVQR. Q and R have been explicitly computed.

3.6.5 The QR Factorization of a Hessenberg Matrix Using Givens Matrices

From the structure of an upper Hessenberg matrix H , it is easy to see that the **QR factorization of H takes only $O(n^2)$ flops** either by Householder's or Givens' method, compared to $O(n^3)$ procedure for a full matrix.

For example, to find the QR factorization of a Hessenberg matrix A using Givens' method, Algorithm 3.6.1 will be reduced to:

Step 1. For $k = 1, 2, \dots, n-1$ do

Step 1.1 Find c and s such that

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a_{kk} \\ a_{k+1,k} \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix}.$$

Step 1.2 Save the index k

Step 1.3 Update $A : A \equiv J(i, c, s, s)A$.

End.

Step 2. Set $R \equiv A$.

Flop-count. The above process requires about $3n^2$ flops. The count does not include explicit computation of Q .

Numerical Stability. Since the Givens QR factorization method is stable for any arbitrary matrix; it is, in particular, stable for a Hessenberg matrix.

QR Factorization with Column Pivoting.

If A is rank-deficient, then QR factorization can not be used to find a basis for $R(A)$. In this case, one needs to use a modification of the QR factorization process, called *QR factorization with column pivoting*.

We shall not discuss this here. The process finds a permutation matrix P , and the matrices Q and R such that $AP = QR$. The details are given in Golub and Van Loan (1996, pp. 248-250). MATLAB function $[Q, R, P] = QR(A)$ can be used to compute the QR factorization with column pivoting.

Also $[Q, R, E] = QR(A, 0)$ produces an economy size QR factorization in which E is a permutation vector so that $Q^*R = A(:, E)$.

3.7 Orthonormal Bases and Orthogonal Projections Using QR Factorization

The QR factorization of A can be used to compute the orthonormal bases and orthogonal projections associated with the subspaces $R(A)$ and $N(A^T)$.

Let A be $m \times n$, where $m \geq n$ and have full rank..

Suppose $Q^T A = R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$. Partition $Q = (Q_1, Q_2)$, where Q_1 has n columns. Then the columns of Q_1 form an orthonormal basis for $R(A)$. Similarly, the columns of Q_2 form an orthonormal basis for the orthogonal complement of $R(A)$. Thus, the matrix $P_A = Q_1 Q_1^T$ is the orthogonal projection onto $R(A)$ and the matrix $P_A^\perp = Q_2 Q_2^T$ is the projection onto the orthogonal complement of $R(A)$. The above projections can also be computed using the singular value decomposition (see **Section 3.9.2**).

MATLAB Note. MATLAB function *orth*(A) computes the orthonormal basis for $R(A)$.

3.8 The Least-Squares Problem

One of the most important applications of the QR factorization of a matrix A is that it can be effectively used to solve the **least-squares problem**.

The **linear least-squares problem** (LSP) is defined as follows:

Given an $m \times n$ matrix A and a real vector b , find a real vector x such that the function:

$$\| r(x) \|_2 = \| Ax - b \|_2$$

is minimized.

If $m > n$, the problem is called an **overdetermined least-square problem**, If $m < n$, it is called an **underdetermined problem**.

We will discuss here only the overdetermined problem.

Theorem 3.8.1 (Theorem on Existence and Uniqueness of the Least Squares problem). *The least-squares solution to $Ax = b$ always exists. The solution is unique if and only if A has full rank. Otherwise, it has infinitely many solutions. The unique solution x is obtained by solving $A^T A x = A^T b$.*

Proof: See Datta (1995, pp. 318).

3.8.1 Solving the Least-Squares Problem Using Normal Equations

The expression of the unique solution in Theorem 3.8.1 immediately suggests the following procedure, called the **Normal Equations** method, for solving the LSP:

1. Compute the **symmetric positive definite matrix** $A^T A$ (Note that if A has full rank, $A^T A$ is symmetric positive definite)
2. Solve for $x : A^T A x = A^T b$.

Computational Remarks. The above procedure, though simple to understand and implement, has **serious numerical difficulties**. **First**, some significant figures may be lost during the explicit formation of the matrix $A^T A$. **Second, the matrix $A^T A$ will be more ill-conditioned, if A is ill-conditioned.** In fact, it can be shown that $\text{Cond}_2(A^T A) = (\text{Cond}_2(A))^2$. The following simple example illustrates the point.

Let

$$A = \begin{pmatrix} 1 & 1 \\ 10^{-4} & 0 \\ 0 & 10^{-4} \end{pmatrix}.$$

If 8-digit arithmetic is used, then $A^T A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, which is **singular**, though the columns of A are linearly independent.

A computationally effective method via the QR factorization of A is now presented below.

3.8.2 Solving the Least-Squares Problem Using QR Factorization

Let $Q^T A = R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$ be the QR decomposition of the matrix A . Then, since the length of a vector is preserved by an orthogonal matrix multiplication, we have

$$\begin{aligned} \|Ax - b\|_2^2 &= \|Q^T Ax - Q^T b\|_2^2 \\ &= \|R_1 x - c\|_2^2 + \|d\|_2^2, \text{ where } Q^T b = \begin{pmatrix} c \\ d \end{pmatrix}. \end{aligned}$$

Thus, $\|Ax - b\|_2^2$ will be minimized if x is chosen so that $R_1 x - c = 0$. The corresponding residual norm then is given by $\|r\|_2 = \|Ax - b\|_2 = \|d\|_2$. This observation immediately suggests the following QR algorithm for solving the least squares problem:

Algorithm 3.8.1 Least Squares Solution Using QR Factorization of A

Inputs:

A - An $m \times n$ matrix ($m \geq n$)

b - An m -vector

Output: The least-squares solution x to the linear system $Ax = b$.

Step 1. Decompose $A = QR$, where $Q \in \mathbb{R}^{m \times m}$ and $R \in \mathbb{R}^{m \times n}$.

Step 2. Form $Q^T b = \begin{pmatrix} c \\ d \end{pmatrix}$, $c \in \mathbb{R}^{n \times 1}$.

Step 3. Obtain x by solving the upper triangular system: $R_1 x = c$ where $R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$.

Step 4. Obtain the residual norm: $\|r\|_2 = \|d\|_2$.

Example 3.8.1 Solve $Ax = b$ for x with

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 5 \\ 9 \end{pmatrix}$$

Step 1. Find the QR Factorization of $A : A = QR$

$$Q = \begin{pmatrix} -0.2673 & 0.8729 & 0.4082 \\ -0.5345 & 0.2182 & -0.8165 \\ -0.8018 & -0.4364 & 0.4082 \end{pmatrix}, R = \begin{pmatrix} -3.7417 & -5.3452 \\ 0 & 0.6547 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

Step 2. Form

$$Q^T b = \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} -10.6904 \\ -0.2182 \\ 0.8165 \end{pmatrix}$$

Step 3. Obtain x by solving $R_1 x = c : x = \begin{pmatrix} 3.3532 \\ -0.3333 \end{pmatrix}$

Step 4. $\|r\|_2 = \|d\|_2 = 0.8165$.

Use of Householder Matrices

Note that if the Householder or Givens' method is used to compute the QR decomposition of A , then the product $Q^T b$ can be formed from the factored form of Q without explicitly computing the matrix Q .

MATCOM and MATLAB Notes: MATCOM function **lsqrqrh** implements the QR factorization method for the full-rank least-squares problem using Householder's method. Alternatively, one can use the MATLAB operator: \backslash . The command $x = A \backslash b$ gives the least-squares solution to $Ax = b$.

Flop-count and Numerical Stability: The least-squares method, using Householder's QR factorization, requires about $2(mn^2 - \frac{n^3}{3})$ flops. The algorithm is numerically stable in the sense that the computed solution satisfies a “nearby” least-squares problem.

3.9 The Singular Value Decomposition (SVD)

We have seen two factorizations (decompositions) of A : LU and QR.

In this section we shall study another important decomposition, called the **singular value decomposition** or in short the **SVD** of A .

Theorem 3.9.1 (The Singular Value Decomposition Theorem).

Given $A \in \mathbb{R}^{m \times n}$, there exist orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$, and a diagonal matrix $\Sigma \in \mathbb{R}^{m \times n}$ with nonnegative diagonal entries such that

$$A = U \Sigma V^T.$$

Proof: See, Datta (1995, pp. 552-554).

The diagonal entries of Σ are called the **singular values** of A .

The columns of U are called the **left singular vectors**, and those of V are called the **right singular vectors**. The singular values are unique, but U and V are not unique.

The number of nonzero singular values is equal to the rank of the matrix A .

A Convention. Let $m \geq n$. Then the n singular values $\sigma_1, \sigma_2, \dots, \sigma_n$ of A can be arranged in nondecreasing order: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$. The largest singular value σ_1 is denoted by σ_{max} . Similarly, the smallest singular value σ_n is denoted by σ_{min} . Since $m \geq n$ is the case mostly arising in applications, we **will assume throughout this section that $m \geq n$** .

The Thin SVD:

Let $U = (u_1, \dots, u_m)$.

If $A = U\Sigma V^T$ be the SVD of $A \in \mathbb{R}^{m \times n}$ and $m \geq n$, and if $U_1 = (u_1, \dots, u_n) \in \mathbb{R}^{m \times n}$, $\Sigma_1 = diag(\sigma_1, \dots, \sigma_n)$, then $A = U_1\Sigma_1V^T$.

This factorization is known as the **thin SVD of A** . For obvious reasons, the thin SVD is also referred to as the **economic SVD**.

Relationship between Eigenvalues and Singular Values

It can be shown that (see Datta (1995), pp. 555-557)

- (i) The singular values $\sigma_1, \dots, \sigma_n$ of A are the nonnegative square roots of the eigenvalues of the symmetric positive semidefinite matrix $A^T A$.
- (ii) The right singular vectors are the eigenvectors of the matrix $A^T A$, and the left singular vectors are the eigenvectors of the matrix AA^T .

Sensitivity of the singular Values

A remarkable property of the singular values is that **they are insensitive to small perturbations**. In other words, the **singular values are well-conditioned**. Specifically, the following result holds.

Theorem 3.9.2 (Insensitivity of the Singular Values)

Let A be an $m \times n$ ($m \geq n$) matrix with the singular values $\sigma_1, \dots, \sigma_n$, and $B = A + E$ be another slightly perturbed matrix with the singular values $\bar{\sigma}_1, \dots, \bar{\sigma}_n$, then $|\bar{\sigma}_i - \sigma_i| \leq \|E\|_2$, $i = 1, \dots, n$.

Proof: See Datta (1995, pp. 560-561).

Example 3.9.1 Let

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}, \quad E = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \times 10^{-4} \end{pmatrix}.$$

The singular values of A :

$$\sigma_1 = 14.5576, \sigma_2 = 1.0372, \sigma_3 = 0$$

The singular values of $A + E$:

$$\bar{\sigma}_1 = 14.5577, \bar{\sigma}_2 = 1.0372, \bar{\sigma}_3 = 2.6492 \times 10^{-5}$$

It is easily verified that the inequalities in the above theorem are satisfied.

3.9.1 The Singular Value Decomposition and the Structure of a Matrix

The SVD is an effective tool in handling several computationally sensitive computations, such as, the **rank** and **rank-deficiency** of matrix, the **distance of a matrix from a matrix of immediate lower rank**, the **orthonormal basis and projections**, etc. It is also a reliable and numerically stable way of computing the least-squares solution to a linear system. Since these computations need to be performed routinely in control and systems theory, we now discuss them briefly in the following.

Theorem 3.9.3 Let $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$ be the n singular values of an $m \times n$ matrix A ($m \geq n$). Then

1. $\|A\|_2 = \sigma_1 = \sigma_{\max}$,
2. $\|A\|_F = (\sigma_1^2 + \sigma_2^2 + \dots + \sigma_n^2)^{\frac{1}{2}}$,
3. $\|A^{-1}\|_2 = \frac{1}{\sigma_n}$, when A is $n \times n$ and nonsingular,
4. $\text{Cond}_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_1}{\sigma_n} = \frac{\sigma_{\max}}{\sigma_{\min}}$, when A is $n \times n$ and nonsingular.

Proof:

1. $\|A\|_2 = \|U\Sigma V^T\|_2 = \|\Sigma\|_2 = \max_i(\sigma_i) = \sigma_1$.

(Note that the 2-norm and F -norm are invariant under multiplication by unitary matrices.)

2. $\|A\|_F = \|U\Sigma V^T\|_F = \|\Sigma\|_F = (\sigma_1^2 + \sigma_2^2 + \dots + \sigma_n^2)^{\frac{1}{2}}$.

3. To prove 3 we note that the largest singular value of A^{-1} is $\frac{1}{\sigma_n}$. (Note that when A is invertible, $\sigma_n \neq 0$). Then the result follows from 1.

4. Follows from the definition of $\text{Cond}_2(A)$ and 1 and 3.

The Condition Number of a Rectangular Matrix

The condition number (with respect to 2-norm) of a rectangular matrix A of order $m \times n$ ($m \geq n$) with full rank is defined to be

$$\text{Cond}_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)},$$

where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ denote, respectively, the largest and smallest singular value of A .

Remark: When A is rank-deficient, $\sigma_{\min} = 0$, and we say that $\text{Cond}(A)$ is **infinite**.

3.9.2 Orthonormal Bases and Orthogonal Projections

Let r be the rank of A , that is,

$$\begin{aligned}\sigma_1 &\geq \sigma_2 \geq \cdots \geq \sigma_r > 0, \\ \sigma_{r+1} &= \cdots = \sigma_n = 0.\end{aligned}$$

Let u_j and v_j be the j th columns of U and V in the SVD of A . Then the set of columns $\{v_j\}$ corresponding to the zero singular values of A form an orthonormal basis for the null-space of A . This is because, when $\sigma_j = 0$, v_j satisfies $Av_j = 0$ and is therefore in the null-space of A . Similarly, the set of columns $\{u_j\}$ corresponding to the nonzero singular values is an orthonormal basis for the range of A . The orthogonal projections now can be easily computed.

Orthogonal Projections

Partition U and V as

$$U = (U_1, U_2), \quad V = (V_1, V_2),$$

where U_1 and V_1 consist of the first r columns of U and V , then

1. **Projection onto $R(A) = U_1 U_1^T$.**
2. **Projection onto $N(A) = V_2 V_2^T$.**
3. **Projection onto the orthogonal complement of $R(A) = U_2 U_2^T$.**
4. **Projection onto the orthogonal complement of $N(A) = V_1 V_1^T$.**

Example 3.9.2

$$\begin{aligned}A &= \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix} \\ \sigma_1 &= 14.5576, \quad \sigma_2 = 1.0372 \quad \sigma_3 = 0.\end{aligned}$$

$$\begin{aligned}U &= \begin{pmatrix} 0.2500 & 0.8371 & 0.4867 \\ 0.4852 & 0.3267 & -0.8111 \\ 0.8378 & -0.4379 & 0.3244 \end{pmatrix} \\ V &= \begin{pmatrix} 0.4625 & -0.7870 & 0.4082 \\ 0.5706 & -0.0882 & -0.8165 \\ 0.6786 & -0.6106 & 0.4082 \end{pmatrix}\end{aligned}$$

An orthonormal basis for the null-space of A is: $V_2 = \begin{bmatrix} 0.4082 \\ -0.8165 \\ 0.4082 \end{bmatrix}$

An orthonormal basis for the range of A is: $U_1 = \begin{bmatrix} 0.2500 & 0.8371 \\ 0.4852 & 0.3267 \\ 0.8370 & -0.4379 \end{bmatrix}$

(Compute now the four orthogonal projections yourself.)

3.9.3 The Rank and the Rank-Deficiency of a Matrix

The most obvious and the least expensive way of determining the rank of a matrix is, of course, to triangularize the matrix using Gaussian elimination and then to find the rank of the reduced upper triangular matrix. Finding the rank of a triangular matrix is trivial; one can just read it off from the diagonal. Unfortunately, however, this is not a very reliable approach in floating point arithmetic. Gaussian elimination method which uses elementary transformations, may transform a rank-deficient matrix into one having full rank, due to numerical round-off error. Thus, in practice, it is more important, **to determine if the given matrix is near a matrix of a certain rank and in particular, to know if it is near a rank-deficient matrix.** **The most reliable way to determine the rank and nearness to rank-deficiency is to use the SVD.**

Suppose that A has rank r , that is, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ and $\sigma_{r+1} = \dots = \sigma_n = 0$. Then the question is: *How far A is from a matrix of rank $k < r$.* The following theorem can be used to answer the question. We state the theorem below, without proof. For proof, see Datta (1995, 565-566).

Theorem 3.9.4 (Distance to Rank-Deficient Matrices). *Let $A = U\Sigma V^T$ be the SVD of A , and let $\text{rank}(A) = r > 0$. Let $k < r$. Define $A_k = U\Sigma_k V^T$, where*

$$\Sigma_k = \left(\begin{array}{ccc|c} \sigma_1 & & 0 & 0 \\ & \ddots & & 0 \\ 0 & & \sigma_k & 0 \\ \hline & & 0 & 0 \end{array} \right), (\sigma_1 \geq \sigma_2, \dots, \geq \sigma_k > 0).$$

1. *Then out of all the matrices of rank k ($k < r$), the matrix A_k is closest to A .*
2. *Furthermore, the distance of A_k from A : $\|A - A_k\|_2 = \sigma_{k+1}$.*

Corollary 3.9.1 *The relative distance of a nonsingular matrix A to the nearest singular matrix B is $\frac{1}{\text{Cond}_2(A)}$. That is, $\frac{\|B - A\|_2}{\|A\|_2} = \frac{1}{\text{Cond}_2(A)}$.*

Implication of the above results

Distance of a Matrix to the Nearest Matrix of Lower Rank

The above result states that the smallest nonzero singular value of A gives the 2-norm distance of A to the nearest matrix of lower rank. In particular, for a nonsingular $n \times n$ matrix A , σ_n gives the measure of the distance of A to the nearest singular matrix.

Thus, in order to know if a matrix A of rank r is close enough to a matrix of lower rank, look into the smallest nonzero singular value σ_r . If this is very small, then the matrix is very close to a matrix of rank $r - 1$, because there exists a perturbation of size as small as $|\sigma_r|$ which will produce a matrix of rank $r - 1$. In fact, one such perturbation is $u_r \sigma_r v_r^T$.

Example 3.9.3 Let

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \times 10^{-7} \end{pmatrix}$$

$$\text{rank}(A) = 3, \quad \sigma_3 = 0.0000004, \quad u_3 = v_3 = (0, 0, 1)^T.$$

$$A' = A - u_3 \sigma_3 v_3^T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

$$\text{rank}(A') = 2.$$

The required perturbation $u_3 \sigma_3 v_3^T$ to make A singular is: $10^{-7} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix}$.

3.9.4 Numerical Rank

The above discussions prompt us to define the concept of “**Numerical Rank**” of a matrix. A has “**numerical rank**” r if the computed singular values $\tilde{\sigma}_1, \tilde{\sigma}_2, \dots, \tilde{\sigma}_n$ satisfy

$$\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \dots \geq \tilde{\sigma}_r > \delta \geq \tilde{\sigma}_{r+1} \geq \dots \geq \tilde{\sigma}_n, \quad (3.9.1)$$

where δ is an error tolerance.

Thus to determine the numerical rank of a matrix A , count the “large” singular values only. If this number is r , then A has numerical rank r .

Remark: Note that finding the numerical rank of a matrix will be “tricky” if there is no suitable gap between a set of singular values.

3.9.5 Solving the Least Squares Problem Using the Singular Value Decomposition

The SVD is also an **effective tool** to solve the least squares problem, both in the full rank and rank-deficient cases.

Recall that the **linear least squares problem** is: Find x such that $\|r\|_2 = \|Ax - b\|_2$ is minimum.

Let $A = U\Sigma V^T$ be the SVD of A . Then since U is orthogonal and a vector length is preserved by orthogonal multiplication, we have

$$\|r\|_2 = \|(U\Sigma V^T x - b)\|_2 = \|U(\Sigma V^T x - U^T b)\|_2 = \|\Sigma y - b'\|_2,$$

where $V^T x = y$ and $U^T b = b'$. Thus, the use of the SVD of A reduces the least squares problem for a full matrix A to one with a diagonal matrix Σ , which is almost trivial to solve, as shown in the following algorithm.

Algorithm 3.9.1 Least Squares Solutions Using the SVD

Inputs: A - An $m \times n$ matrix,

b - An m -vector

Output: x - The least-squares solution of the system $Ax = b$.

Step 1. Find the SVD of A : $A = U\Sigma V^T$.

Step 2. Form $b' = U^T b = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_m \end{pmatrix}$.

Step 3. Compute $y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ choosing $y_i = \begin{cases} \frac{b'_i}{\sigma_i}, & \text{when } \sigma_i \neq 0 \\ \text{arbitrary,} & \text{when } \sigma_i = 0. \end{cases}$.

Step 4. Compute the family of least squares solutions: $x = Vy$. (Note that in the full-rank case, the family has just one number.)

Flop-Count. Using the SVD, it takes about $4mn^2 + 8n^3$ flops to solve the least squares problem, when A is $m \times n$ and $m \geq n$.

An Expression for the Minimum Norm Least Squares Solution

Since a rank-deficient least-squares problem has an infinite number of solutions, it is practical to look for the one that has minimum norm. Such a solution is called the **minimum norm least square solution**.

It is clear from Step 3 above that in the **rank-deficient case** the minimum 2-norm least squares solution is the one that is obtained by setting $y_i = 0$, whenever $\sigma_i = 0$. Thus, from above, we have the following expression for the minimum 2-norm solution:

$$x = \sum_{i=1}^k \frac{u_i^T b'_i}{\sigma_i} v_i, \quad (3.9.1)$$

where k is the **numerical rank** of A , and u_i and v_i , respectively, are the i th columns of U and V .

Example 3.9.4

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 1 & 2 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} 6 \\ 9 \\ 6 \end{pmatrix}.$$

Step 1. $\sigma_1 = 7.5358$, $\sigma_2 = 0.4597$, $\sigma_3 = 0$.

A is rank-deficient.

$$U = \begin{pmatrix} 0.4956 & 0.5044 & 0.7071 \\ 0.7133 & -0.7008 & 0.0000 \\ 0.4956 & 0.5044 & -0.7071 \end{pmatrix}, \quad V = \begin{pmatrix} 0.3208 & -0.8546 & 0.4082 \\ 0.5470 & -0.1847 & -0.8165 \\ 0.7732 & 0.4853 & 0.4082 \end{pmatrix}.$$

Step 2. $b' = U^T b = (12.3667, -0.2547, 0)^T$.

Step 3. $y = (1.6411, -0.5541, 0)$.

The minimum 2-norm least-squares solution is $Vy = (1, 1, 1)^T$.

Computing the SVD of A

Since the singular values of a matrix A are the nonnegative square roots of the eigenvalues of $A^T A$, it is natural to think of computing the singular values and the singular vectors, by finding the eigendecomposition of $A^T A$. However, **this is not a numerically effective procedure**. Some vital information may be lost during the formation of the matrix $A^T A$, as the following example shows.

$$\text{Example 3.9.5 } A = \begin{pmatrix} 1.0001 & 1.000 \\ 1.000 & 1.0001 \end{pmatrix}.$$

The singular values of A are 2.0001 and 0.0001.

$A^T A = \begin{pmatrix} 2.0002 & 2.0002 \\ 2.0002 & 2.0002 \end{pmatrix}$. The eigenvalues of $A^T A$ are 0 and 4.0008 (in four digit arithmetic). Thus, the singular values computed from the eigenvalues of $A^T A$ are 0 and 2.0002. A standard algorithm for computing the SVD of A is the **Golub-Kahan-Reinsch** algorithm. The algorithm will be described later in the book in Chapter 4.

MATLAB and MATCOM Notes: MATLAB function **svd** can be used to compute the SVD. $[U, S, V] = \text{svd}(A)$ produces a diagonal matrix S , of the same dimension as A and with nonnegative diagonal entries in decreasing order, and unitary matrices U and V such that $A = USV^*$.

Algorithm 3.9.1 has been implemented in MATCOM program **lsqrsvd**. Also, MATCOM has a program called **minmsvd** to compute the minimum 2-norm least-squares solution using the SVD.

3.10 Summary and Review

I. Floating Point Numbers and Errors

1. *Floating-point numbers:* A t -digit floating point number has the form

$$x = m\beta^e$$

where e is called exponent, m is a t -digit fraction and β is the base of the number system.

2. *Errors:* The errors in a computation are measured either by absolute error or relative error. **The relative errors make more sense than absolute errors.** The relative error gives an indication of the number of significant digits in an approximate answer. The relative error in representing a real number x by its floating-point representation $fl(x)$ is bounded by a number μ , called the **machine precision** (**Theorem 3.2.1**).

3. *Laws of floating-point arithmetic:*

$$fl(x * y) = (x * y)(1 + \delta)$$

II. Conditioning, Stability and Accuracy

1. *Conditioning of the problem:* The conditioning of the problem is a property of the problem. A problem is said to be **ill conditioned** if a small change in the data causes a large change in the solution, otherwise it is **well conditioned**. The conditioning of a problem is data dependent. A problem can be ill conditioned with respect to one set of data but can be quite well conditioned with respect to another set.

The condition number of a nonsingular matrix, $\text{Cond}(A) = \|A\| \|A^{-1}\|$ is an indicator of the conditioning of the associated linear system problem: $Ax = b$. If $\text{Cond}(A)$ is large, then the linear system $Ax = b$ is ill-conditioned.

The well-known examples of ill-conditioned problems are the **Wilkinson polynomial** for the root-finding problem, the **Wilkinson bidiagonal matrix** for the eigenvalue problem, the **Hilbert matrix** for the algebraic linear system problem, and so on.

2. *Stability of an algorithm:* An algorithm is said to be a *backward stable algorithm* if it computes the exact solution of a nearby problem. Some examples of stable algorithms are the methods of back substitution and forward elimination for triangular systems, the QR factorization using Householder and Givens matrices transformations, and the QR iteration algorithm for eigenvalue computations.

The Gaussian elimination algorithm without row changes is unstable for arbitrary matrices. However, Gaussian elimination with partial pivoting can be considered as a stable algorithm in practice.

3. *Effects of conditioning and stability on the accuracy of the solution:* The conditioning of the problem and the stability of the algorithm both have effects on accuracy of the solution computed by the algorithm.

If a stable algorithm is applied to a well-conditioned problem, it should compute an accurate solution. On the other hand, if a stable algorithm is applied to an ill-conditioned problem, there is no guarantee that the computed solution will be accurate. However, if a stable algorithm is applied to an ill-conditioned problem, it should not introduce more errors than that which the data warrants.

III. Matrix Factorizations. There are three important matrix factorizations: *LU*, *QR*, and *SVD*.

1. *LU Factorization.* A factorization of a matrix A in the form $A = LU$, where L is unit lower triangular and U is upper triangular, is called an *LU* factorization of A . An *LU* factorization of A exists if all of its leading principal minors are nonsingular. A classical elimination scheme, called **Gaussian elimination**, is used to obtain an *LU* factorization of A (**Section 3.4.1**).

The stability of Gaussian elimination is determined by the **growth factor**

$$\rho = \frac{\max(\alpha, \alpha_1, \dots, \alpha_{n-1})}{\alpha}, \text{ where } \alpha = \max_{i,j} |a_{ij}| \text{ and } \alpha_k = \max_{i,j} |a_{ij}^{(k)}|.$$

If no pivoting is used in Gaussian elimination, ρ can be arbitrarily large. Thus, **Gaussian elimination without pivoting is, in general, an unstable process**.

If partial pivoting is used, then Gaussian elimination yields the factorization of A in the form $PA = LU$, where P is a perturbation matrix.

The growth factor ρ for Gaussian elimination with partial pivoting can be as large as 2^{n-1} ; however, such a growth is extremely rare in practice. Thus, **Gaussian elimination with partial pivoting is considered to be a stable process in practice.**

2. *The QR Factorization.* Given an $m \times n$ matrix, there exists an orthogonal matrix Q and an upper triangular matrix R such that $Q = QR$.

The QR factorization of A can be obtained using **Householder's method**, **Givens' method**, **The Gram-Schmidt Processes** (The CGS and MGS).

The Gram-Schmidt processes do not have favorable numerical properties. **Both Householder's and Givens' methods are numerically stable procedures for QR factorization.** They are discussed, respectively, in **Section 3.6.2** and **Section 3.6.4.** (**Algorithm 3.6.1**). Householder's method is slightly more efficient than Givens' method.

IV. The Algebraic Linear System Problem $Ax = b$.

The method of practical choice for the linear system problem $Ax = b$ is Gaussian elimination with partial pivoting (**Section 3.5.2**) followed by iterative refinement procedure (**Section 3.5.7**). A symmetric positive definite system should be solved by computing its Cholesky factor (**Algorithm 3.4.1**) R followed by solving two triangular systems: $Ry = b$ and $R^T x = y$ (**Algorithm 3.3.1** and **3.3.2**).

- V. **The Least-Squares Problem (LSP).** Given an $m \times n$ matrix A , the LSP is the problem of finding a vector x such that $\|Ax - b\|_2$ is minimized. The LSP can be solved using

- The Normal Equations Method (**Section 3.8**): $A^T Ax = A^T b$
- The QR Factorization Method (**Algorithm 3.8.1**)
- The SVD Method (**Algorithm 3.9.1**)

The normal equations method might give numerical difficulties, and should not be used in practice without looking closely at the condition number. Both the QR and SVD methods for the LSP are numerically stable. Though the SVD is more expensive than the QR method, **the SVD method is most reliable and can handle both rank-deficient and full-rank cases very effectively.**

VI. The Singular Value Decomposition

1. *Existence and uniqueness of the SVD:* The singular value decomposition (SVD) of a matrix A always exists (**Theorem 3.9.1**):

Let $A \in \mathbb{R}^{m \times n}$. Then $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ are orthogonal and Σ is an $m \times n$ diagonal matrix.

The singular values (the diagonal entries of Σ) are unique, but U and V are not unique.

2. *Relationship between the singular values and the eigenvalues:* The singular values of A are the nonnegative square roots of the eigenvalues of $A^T A$ (or of AA^T).
3. *Sensitivity of the singular values:* The singular values are insensitive to small perturbations (**Theorem 3.9.2**)
4. *Applications of the SVD:* The singular values and the singular vectors of a matrix A are useful and are the most reliable tools for determining the (numerical) rank and the rank-deficiency of A ; finding the orthonormal bases for range and the null space of A ; finding the distance of A from another matrix of lower rank (in particular the nearness to singularity of a nonsingular matrix); solving both full-rank and the rank-deficient least-squares problems.

These remarkable abilities and the fact that the singular values are insensitive to small perturbations have made the SVD an indispensable tool for a wide variety of problems in control and systems theory.

3.11 Chapter Notes and Further Reading

Material of this chapter has been taken from the recent book of the author (Datta (1995)). For the advanced topics on numerical linear algebra, see Golub and Van Loan (1996). The details about stability of various algorithm and sensitivities of problems described in this chapter can be found in the book by Higham (1996). Stewart's recent book (Stewart (1998)) is also an excellent source of knowledge in this area. For details of various MATLAB functions, see MATLAB Users' Guide (1992). *MATCOM* is a MATLAB-based toolbox implementing all the major algorithms of the book "*Numerical Linear Algebra and Applications*" by Datta (1995). The toolbox can be obtained from the office of MATHWORKS. See the instructions in the above book of how to obtain it.

References

1. B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, Pacific Grove, CA, 1995.
2. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.
3. N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.
4. MATLAB *User's Guide*, The Math Works, Inc., Natick, MA, 1992.
5. G.W. Stewart, *Matrix Algorithms Volume 1: Basic Decompositions*, SIAM, Philadelphia, 1998.
6. J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

Chapter 4

CANONICAL FORMS OBTAINED VIA ORTHOGONAL TRANSFORMATIONS

Contents

4.1	Importance and Significance of Using Orthogonal Transformations	108
4.2	Hessenberg Reduction of a Matrix	110
4.2.1	Uniqueness in Hessenberg Reduction: The Implicit Q Theorem	111
4.3	The Real Schur Form of A : The QR Iteration Method	112
4.3.1	The Basic QR Iteration	113
4.3.2	The Hessenberg QR Iteration and Shift of Origin	113
4.3.3	The Double Shift QR Iteration	114
4.3.4	Obtaining the Real Schur Form A	115
4.3.5	The Real Schur Form and Invariant Subspaces	117
4.4	Power Iteration and Inverse Iteration	119
4.5	Computing the Singular Value Decomposition	120
4.6	The Generalized Real Schur Form: The QZ algorithm	122
4.6.1	Reduction to Hessenberg-Triangular Form	124
4.6.2	Reduction to the Generalized Real Schur Form	126
4.7	Computing of the Eigenvectors of the Pencil $A - \lambda B$	128
4.8	Summary and Review	129
4.9	Chapter Notes and Further Reading	131

Topics Covered

- Numerical Instabilities in obtaining the Jordan and Companion Matrices
- Hessenberg Reduction of a Matrix
- The Double-Shift Implicit QR Iteration for the Real Schur Form of a Matrix
- Invariant Subspace Computation from the Real Schur Form
- The QZ Algorithm for the Generalized Real Schur Form of the Matrix Pencil $A - \lambda B$
- The SVD Computation

4.1 Importance and Significance of Using Orthogonal Transformations

The Jordan and companion matrices have special structures that can be conveniently exploited to solve many control problems. **Unfortunately, however, these forms in general, cannot be obtained in a numerically stable way.**

We examine this fact here in some details below.

Suppose that X is a nonsingular matrix and consider the computation of $X^{-1}AX$ in floating point arithmetic. It can be shown that

$$fl(X^{-1}AX) = X^{-1}AX + E,$$

where $\|E\|_2 \approx \mu \text{Cond}(X) \|A\|_2$, μ is the machine precision.

Thus, when X is ill-conditioned, there will be large errors in computing $X^{-1}AX$.

For the Jordan canonical form, the transforming matrix X is highly ill-conditioned, whenever A has defective or nearly defective eigenvalue.

The reduction of a matrix A to a companion matrix C of the form

$$C = \begin{pmatrix} 0 & 0 & \cdots & 0 & c_1 \\ 1 & 0 & \cdots & 0 & c_2 \\ 0 & 1 & \cdots & 0 & c_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_n \end{pmatrix}$$

involves the following steps:

Step 1. A is transformed to an upper Hessenberg matrix $H_u = (h_{ij})$ by orthogonal similarity:
 $P^T AP = H_u$.

Step 2. Assuming that H_u is unreduced, that is $h_{i+1,i} \neq 0, i = 1, 2, \dots, n-1$, then H_u is further reduced to the companion matrix C by similarity. Thus, if $Y = (e_1, H_ue_1, \dots, H_u^{n-1}e_1)$, then it is easy to see that $Y^{-1}H_uY = C$.

A numerically stable algorithm to implement Step 1 is given in the next section; however, the matrix Y in Step 2 **can be highly ill-conditioned if H_u has small subdiagonal entries.** (Note that Y is a lower triangular matrix with $1, h_{21}h_{32}, \dots, h_{21}h_{32} \dots h_{n,n-1}$ as the diagonal entries).

Thus, Step 2, in general, cannot be implemented in a numerically effective manner. The above discussions clearly show that it is important from a numerical computation viewpoint to have canonical forms which can be achieved using only well-conditioned transforming matrices, such as orthogonal matrices.

Indeed, **if a matrix A is transformed to a matrix B using an orthogonal similarity transformation, then a perturbation in A will result in a perturbation in B of the same magnitude.** That is, if

$$B = U^T AU$$

and

$$U^T(A + \Delta A)U = B + \Delta B,$$

then

$$\|\Delta B\|_2 \approx \|\Delta A\|_2.$$

In this Chapter, we show that two very important canonical forms: the **Hessenberg form** and the **Real Schur Form** (RSF) of a matrix A , can be obtained using orthogonal similarity transformations. (Another important canonical form, known as the **generalized real Schur form**, can be obtained using orthogonal equivalence.)

We will see in the remaining of the book that **these canonical forms form important tools in the development of numerically effective algorithms for control problems.**

Applications of **Hessenberg** and **real Schur** forms include

- (i) Computation of frequency response matrix (**Chapter 5**)
- (ii) Solutions of Lyapunov and Sylvester equations (**Chapter 8**), Algebraic Riccati equations (**Chapter 13**), Sylvester-observer equation (**Chapter 12**).
- (iii) Solutions of eigenvalue assignment (**Chapter 11**), feedback stabilization (**Chapter 10**) problems, stability and inertia computations (**Chapter 7**).

Applications of generalized real Schur form include

- (i) Solutions of certain algebraic Riccati equations (**Chapter 13**)

- (ii) Solution of any descriptor control problem
- (iii) Computations of frequencies and modes of vibrating systems (**Chapter 16**).

Besides these two forms, there are two other important canonical forms; namely the **Controller-Hessenberg** and **observer-Hessenberg** forms, which also can be obtained in a numerically effective way, will be used throughout the book. Methods for obtaining these two forms are described in **Chapter 6**.

4.2 Hessenberg Reduction of a Matrix

Recall that a matrix $H = (h_{ij})$ is said to be an **upper Hessenberg** matrix if $h_{ij} = 0$ for $i > j + 1$.

An $n \times n$ matrix A can always be transformed to an upper Hessenberg matrix H_u by orthogonal similarity. That is, **given an $n \times n$ matrix A , there exists an orthogonal matrix P such that $PAP^T = H_u$** .

Again, Householder and Givens matrices, being orthogonal, can be employed to obtain H_u from A .

We will discuss only Householder's method here.

Reduction to Hessenberg Form using Householder Matrices

The idea is to extend the QR factorization process using Householder matrices described in Chapter 3 to obtain P and H_u , such that $PAP^T = H_u$ is an upper Hessenberg matrix and P is orthogonal.

The matrix P is constructed as the product of $(n - 2)$ Householder matrices P_1 through P_{n-2} . The matrix P_1 is constructed to create zeros in the first column of A below the entry (2,1); P_2 is constructed to create zeros below the entry (3,2) of the second column of the matrix $P_1AP_1^T$, and so on.

The process consists of $(n - 2)$ steps. (Note that an $n \times n$ Hessenberg matrix contains at least $\frac{(n-2)(n-1)}{2}$ zeros.)

At the end of $(n - 2)th$ step, the matrix $A^{(n-2)}$ is an upper Hessenberg matrix H_u . The Hessenberg matrix H_u is orthogonally similar to A . This is seen as follows:

$$\begin{aligned} H_u &= A^{(n-2)} = P_{n-2}A^{(n-3)}P_{n-2}^T = P_{n-2}(P_{n-3}A^{(n-4)}P_{n-3}^T)P_{n-2}^T \\ &\quad \cdots = (P_{n-2}P_{n-3} \cdots P_1)A(P_1^TP_2^T \cdots P_{n-3}^TP_{n-2}^T). \end{aligned} \tag{4.2.1}$$

Set

$$P = P_{n-2}P_{n-3} \cdots P_1. \tag{4.2.2}$$

We then have $H_u = PAP^T$. Since each Householder matrix P_i is orthogonal, the matrix P which is the product of $(n - 2)$ Householder matrices, is also orthogonal.

For $n = 4$, schematically, we can represent the reduction as follow. Set $A^{(0)} \equiv A$. Then

$$A \xrightarrow{P_1} P_1 A P_1^T = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} = A^{(1)}$$

$$A^{(1)} \xrightarrow{P_2} P_2 A^{(1)} P_2^T = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{pmatrix} = A^{(2)} = H_u$$

- Notes:**
1. Multiplication by P_i^T to the right does not destroy the zeros already present in $P_i A^{(i-1)}$.
 2. The product $P_i A^{(i-1)} P_i^T$ can be **implicitly** formed as shown in Chapter 3 (**Section 3.6.1**).

Flop count. The process requires $\frac{10}{3}n^3$ flops to compute H_u . This count does not include the explicit computation of P , which is stored in factored form. If P is computed explicitly, another $\frac{4}{3}n^3$ flops are required. However, when n is large, the storage required to form P is prohibitive.

Roundoff property. The process is numerically stable. It can be shown (Wilkinson (1965), p. 351) that the computed H_u is orthogonally similar to a nearby matrix $A + E$, where

$$\|E\|_F \leq cn^2\mu \|A\|_F.$$

Here c is a constant of order unity.

MATLAB NOTE: The MATLAB Command $[P, H] = \text{hess}(A)$ computes an orthogonal matrix P and an upper Hessenberg matrix H such that $PAP^T = H$.

4.2.1 Uniqueness in Hessenberg Reduction: The Implicit Q Theorem

We just described Householder's method for Hessenberg reduction. However, this form could also have been obtained using **Givens matrices** as well (see Datta (1995), pp. 163-165). The question, therefore, arises **how unique is the Hessenberg form?**

The question is answered in the following theorem, known as the **Implicit Q Theorem**. The proof can be found in Golub and Van Loan (1996, p. 347).

Theorem 4.2.1 (The Implicit Q Theorem) Let $P = (p_1, p_2, \dots, p_n)$ and $Q = (q_1, q_2, \dots, q_n)$ be orthogonal matrices such that $P^T AP = H_1$ and $Q^T AQ = H_2$ are two **unreduced** upper Hessenberg matrices. Suppose that $p_1 = q_1$. Then H_1 and H_2 are essentially the same in the sense that $H_2 = D^{-1}H_1D$, where $D = \text{diag}(\pm 1, \dots, \pm 1)$. Furthermore, $p_i = \pm q_i, i = 2, \dots, n$.

4.3 The Real Schur Form of A : The QR Iteration Method

In this section, we describe how to obtain the **Real Schur Form** of a matrix (RSF). The RSF of a matrix A displays the eigenvalues of A . It is obtained by using the well-known QR iteration method. **This method is nowadays a standard method for computing the eigenvalues of a matrix.** First, we state a well-known classical result on this subject.

Theorem 4.3.1 (The Schur Triangularization Theorem). *Let A be an $n \times n$ complex matrix, then there exists an $n \times n$ unitary matrix U such that*

$$U^*AU = T$$

where T is an $n \times n$ upper triangular matrix and the diagonal entries of T are the eigenvalues of A .

Proof: See Datta (1995, pp. 433-439).

Since a real matrix can have complex eigenvalues (occurring in complex conjugate pairs), even for a real matrix A , U and T in the above theorem above can be complex. However, we can choose U to be real orthogonal if T is replaced by a **quasi-triangular matrix** R , known as the **Real Schur Form** of A (**RSF**), as the following theorem shows. The proof can be found in Datta (1995, p. 434) or in Golub and Van Loan (1996, pp. 341-342).

Theorem 4.3.2 (The Real Schur Triangularization Theorem) *Let A be an $n \times n$ real matrix. Then there exists an $n \times n$ orthogonal matrix Q such that*

$$Q^T AQ = R = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1k} \\ 0 & R_{22} & \cdots & R_{2k} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & R_{kk} \end{pmatrix} \quad (4.3.3)$$

where each R_{ii} is either a scalar or a 2×2 matrix. The scalars diagonal entries correspond to real eigenvalues, and each 2×2 matrix on the diagonal has a pair of complex conjugate eigenvalues.

Definition 4.3.1 The matrix R in Theorem 4.3.2 is known as the **Real Schur Form** (RSF) of A .

Remarks:

1. The 2×2 matrices on the diagonal are usually referred to as “**Schur bumps**”.
2. The columns of Q are called the **Schur vectors**. For each $k = 1, 2, \dots, n$, the first k columns of Q form an orthonormal basis for the invariant subspace corresponding to the first k eigenvalues.

We present below a method, known as the QR iteration method, for computing the real-Schur form of A . **A properly implemented QR method is widely used nowadays for computing the eigenvalues of an arbitrary matrix.** As the name suggests, the method is based on the QR factorization and is iterative in nature. Since the roots of a polynomial equation of degree higher than four cannot be found in a finite number of steps, **any numerical method to compute the eigenvalues a matrix of order higher than four has to be iterative in nature.** The QR iteration method was proposed in algorithmic form by J. G. Francis (1961), though its roots can be traced to a work of Rutishauser (1958). The method was also independently discovered by the Russian mathematician Kublanovskaya (1961). For references of these papers, see Datta (1995) or Golub and Van Loan (1996).

4.3.1 The Basic QR Iteration

We first present the basic QR iteration method.

$$\text{Set } A_0 \equiv A.$$

Compute now a sequence of matrices $\{A_k\}$ as follows:

For $k = 1, 2, \dots$ do

Find the QR factorization of $A_{k-1} : A_{k-1} = Q_k R_k$

Compute $A_k = R_k Q_k$.

End

The matrices in the sequence $\{A_k\}$ have a very interesting property: **Each matrix in the sequence is orthogonally similar to the previous one and is, therefore, orthogonally similar to the original matrix.** It is easy to see this. For example,

$$\begin{aligned} A_1 &= R_1 Q_1 = Q_1^T A_0 Q_1 (\text{since } R_1 = Q_1^T A_0), \\ A_2 &= R_2 Q_2 = Q_2^T A_1 Q_2 (\text{since } R_2 = Q_2^T A_1). \end{aligned}$$

Thus A_1 is orthogonally similar to A , and A_2 is orthogonally similar to A_1 . Therefore, A_2 is orthogonally similar to A , as the following computation shows:

$$A_2 = Q_2^T A_1 Q_2 = Q_2^T (Q_1^T A_0 Q_1) Q_2 = (Q_1 Q_2)^T A_0 (Q_1 Q_2).$$

Since each matrix A_k is orthogonally similar to the original matrix A , it has the same eigenvalues as A . It can then be shown (Wilkinson (1965), pp. 518-519) that under certain conditions, the sequence $\{A_k\}$ converges to the real Schur form (RSF) or to the Schur form of A .

4.3.2 The Hessenberg QR Iteration and Shift of Origin

The QR iteration method as presented above is not practical if the matrix A is full and dense. This is because, as we have seen before, that the QR factorization of a matrix A requires $O(n^3)$ flops and thus n iterations will consume $O(n^4)$ flops, making the method impractical.

Fortunately, something simple can be done:

Reduce the matrix A to a Hessenberg matrix by orthogonal similarity before starting the QR iterations. An interesting practical consequence of this is that if $A = A_0$ is initially reduced to an upper Hessenberg matrix H and is assumed to be unreduced, then each member of the matrix sequence $\{H_k\}$ obtained by applying QR iteration to H is also upper Hessenberg.

The question now arises of how the reduction to the Hessenberg form helps.

To answer this question, we first note that the reduction of A to a Hessenberg matrix has to be done only once. As we know that this process, using Householder's method requires $\frac{10}{3}n^3$ flops. Furthermore, the QR factorization of a Hessenberg matrix requires only $O(n^2)$ flops (Section 3.6.5). **Since each H_k is Hessenberg, the QR iteration method with the initial reduction of A to a Hessenberg matrix will be an efficient procedure.**

Though the reduction of A to a Hessenberg matrix H helps in the sense that it makes the process a $O(n^3)$ process, the convergence of the subdiagonal entries of H , in the presence of two or more nearly equal (in magnitude) eigenvalues, can be painfully slow.

Fortunately, the rate of convergence can be significantly improved by using a suitable shift.

The idea is to apply the QR iteration to the shifted matrix $\hat{H} = H - \hat{\lambda}_i I$, where $\hat{\lambda}_i$ is an approximate eigenvalue. This is known as the **single shift QR iteration**.

However, since the complex eigenvalues of a real matrix occur in conjugate pairs; in practice, the QR iteration is applied to the matrix H with double shifts. The process then is called the **double shift QR iteration** method.

4.3.3 The Double Shift QR Iteration

The Hessenberg double-shift QR iteration scheme can be written as follows:

For $i = 1, 2, \dots$ do

Choose the two shifts k_1 and k_2

Find the QR Factorization: $H - k_1 I = Q_1 R_1$

Form: $H_1 = R_1 Q_1 + k_1 I$

Find the QR factorization: $H_1 - k_2 I = Q_2 R_2$

Form: $H_2 = R_2 Q_2 + k_2 I$

End

The shifts k_1 and k_2 at each iteration are chosen as the eigenvalues of the 2×2 trailing principal submatrix at that iteration. The process is called the **explicit double-shift QR iteration** process.

The above explicit scheme requires complex arithmetic (since k_1 and k_2 are complex) to implement, and furthermore, the matrices $H - k_1 I$ and $H_1 - k_2 I$ need to be formed explicitly. In

practice, an equivalent implicit version, known as the **double shift implicit QR iteration scheme**, is used. We state one step of this process in the following.

The Double Shift Implicit QR Step

1. Compute the first column n_1 of the matrix $N = (H - k_1 I)(H - k_2 I) = H^2 - (k_1 + k_2)H + k_1 k_2 I$.
2. Find a Householder matrix P_0 such that $P_0 n_1$ is a multiple of e_1 .
3. Find Householder matrices P_1 through P_{n-2} such that $H_2 = (P_{n-2}^T \cdots P_1^T P_0^T)H(P_0 P_1 \cdots P_{n-2})$ is an upper Hessenberg matrix.

It can be shown by using the **Implicit Q Theorem** (Theorem 4.2.1), that **the upper Hessenberg matrix H_2 obtained by the double shift implicit QR step is essentially the same as H_2 obtained by one step of the explicit scheme**. Furthermore, the first column n_1 of N can be computed without explicitly computing the matrix N and, the computation of H_2 from H can be done only in $O(n^2)$ flops. For details see Datta (1995, pp. 444-447).

4.3.4 Obtaining the Real Schur Form A

1. Transform the matrix A to Hessenberg form.
2. Iterate with the double shift implicit QR step.

Typically, after two to three iteration steps of the double shift implicit QR method, one or two (and sometimes more) subdiagonal entries from the bottom of the Hessenberg matrix converge to zero. This then will give us a real or a pair of complex conjugate eigenvalues.

Once a real or a pair of complex conjugate eigenvalues is computed, the last row and the last column in the first case, or the last two rows and the last two columns in the second case, are deleted and the computation of the other eigenvalues is continued with the submatrix.

This process is known as **deflation**.

Note that the eigenvalues of the deflated submatrix are also the eigenvalues of the original matrix. For, suppose, immediately before deflation, the matrix has the form:

$$H_k = \begin{pmatrix} A' & C' \\ 0 & B' \end{pmatrix},$$

where B' is the 2×2 trailing submatrix or is an 1×1 matrix. Then the characteristic polynomial of H_k is: $\det(\lambda I - H_k) = \det(\lambda I - A') \det(\lambda I - B')$. Thus, the eigenvalues of H_k are the eigenvalues of A' together with those of B' . But H_k is orthogonally similar to the original matrix A and therefore has the same eigenvalues as A .

Example 4.3.1 Find the real Schur form of

$$H = \begin{bmatrix} 0.2190 & -0.0756 & 0.6787 & -0.6391 \\ -0.9615 & 0.9032 & -0.4571 & 0.8804 \\ 0 & -0.3822 & 0.4526 & -0.0641 \\ 0 & 0 & -0.1069 & -0.0252 \end{bmatrix}.$$

Iteration	h_{21}	h_{32}	h_{43}
1	0.3860	-0.5084	-0.0084
2	-0.0672	-0.3773	0.0001
3	0.0089	-0.3673	0
4	-0.0011	-0.3590	0
5	0.0001	-0.3905	0
...			

The computed RSF is

$$H = \begin{bmatrix} 1.4095 & 0.7632 & -0.1996 & 0.8394 \\ 0.0001 & \boxed{0.1922} & 0.5792 & 0.0494 \\ 0 & -0.3905 & 0.0243 & -0.4089 \\ 0 & 0 & 0 & -0.0763 \end{bmatrix}.$$

The eigenvalues of $\begin{bmatrix} 0.1922 & 0.5792 \\ -0.3905 & 0.0243 \end{bmatrix}$ are $0.1082 \pm 0.4681j$.

Balancing

It is advisable to balance the entries of the original matrix A , if they vary widely, before starting the QR process.

The balancing is equivalent to transforming the matrix A to $D^{-1}AD$, where the diagonal matrix D is chosen so that the transformed matrix has approximately equal row and column norms. In general, preprocessing the matrix by balancing improves the accuracy of the QR iteration method. **Note that no round-off error is involved in this computation and it takes only $O(n^2)$ flops.**

MATLAB Note: The MATLAB function $[T, B] = \text{balance}(A)$ finds a diagonal matrix T such that $B = T^{-1}AT$ has approximately the equal row and column norms.

Flop-count of the QR Iteration Method: Since the QR iteration method is an iterative method, it is hard to give an exact flop count for this method. However, empirical observations have established that it takes about two QR iterations per eigenvalue. Thus, it will require about $12n^3$ flops to compute all the eigenvalues. If the transforming matrix Q and the final quasitriangular matrix T are also needed, then the cost will be about $26n^3$ flops.

Numerical Stability Property of the QR Iteration Process. The QR iteration method is quite stable. An analysis of the round-off property of the algorithm shows that the computed real Schur form (RSF) \hat{T} is orthogonally similar to a nearby matrix $A + E$. Specifically,

$$Q^T(A + E)Q = \hat{T}, \text{ where } \|E\|_F \leq \phi(n)\mu\|A\|_F,$$

where $\phi(n)$ is a slowly growing function of n and μ is the machine precision. The computed orthogonal matrix Q can also be shown to be nearly orthogonal.

MATLAB Notes: The MATLAB function **schur** in the following format: $[U, T] = \text{schur}(A)$ produces a Schur matrix T and an unitary matrix U such that $A = UTU^*$.

By itself, **schur** (A) returns T . If A is real, the real Schur form is returned.

The real Schur Form has the real eigenvalues on the diagonal and the complex eigenvalues in 2×2 blocks on the diagonal.

4.3.5 The Real Schur Form and Invariant Subspaces

Once again, we remind the readers that **the real Schur form of A displays information on the invariant subspaces.**

Basis of an Invariant Subspace from RSF

Let

$$Q^T A Q = R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

and let's assume that R_{11} and R_{22} do not have eigenvalues in common. Then the first p columns of Q , where p is the order of R_{11} , form a basis for the invariant subspace associated with the eigenvalues of R_{11} .

In many applications, such as in the **solution of algebraic Riccati equations** (see Chapter 13)), in constructing a reduced-order model, etc., one needs to compute an orthonormal basis of an invariant subspace associated with a selected number of eigenvalues. Unfortunately, the real Schur form obtained by QR iteration will not, in general, give the eigenvalues in some desired order. Thus, if the eigenvalues are not in a desired order, one wonders if some extra work can be done to bring them into that order. That this can indeed be done, is seen from the following simple discussion. Let A be 2×2 .

Let

$$Q_1^T A Q_1 = \begin{pmatrix} \lambda_1 & r_{12} \\ 0 & \lambda_2 \end{pmatrix}, \quad \lambda_1 \neq \lambda_2.$$

If λ_1 and λ_2 are not in right order, all we need to do to reverse the order is to form a Givens rotation $J(1, 2, \theta)$ such that

$$J(1, 2, \theta) \begin{pmatrix} r_{12} \\ \lambda_2 - \lambda_1 \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix}.$$

Then $Q = Q_1 J(1, 2, \theta)^T$ is such that

$$Q^T A Q = \begin{pmatrix} \lambda_2 & r_{12} \\ 0 & \lambda_1 \end{pmatrix}.$$

The above simple process can be easily extended to achieve any desired ordering of the eigenvalues in the Real Schur Form. For a Fortran program, see Stewart (1976).

Example 4.3.1

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$$

$$Q_1 = \begin{pmatrix} .8507 & .5257 \\ -.5257 & .8507 \end{pmatrix}$$

$$Q_1^T A Q_1 = \begin{pmatrix} -0.2361 & 0.0000 \\ 0.0000 & 4.2361 \end{pmatrix}$$

Suppose we now want to reverse the orders of -0.2361, and 4.2361.

Form: $J(1, 2, \theta) = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$.

Then $J(1, 2, \theta) \begin{pmatrix} 0 \\ 4.4722 \end{pmatrix} = \begin{pmatrix} 4.4722 \\ 0 \end{pmatrix}$.

Form: $Q = Q_1 J(1, 2, \theta)^T = \begin{pmatrix} -0.5257 & -0.8507 \\ -0.8507 & 0.5257 \end{pmatrix}$.

Then $Q^T A Q = \begin{pmatrix} 4.2361 & 0.00 \\ 0.00 & -0.2361 \end{pmatrix}$.

Flop Count and Numerical Stability. The process is quite inexpensive. It requires only $k(12n)$ flops, where k is the number of interchanges required to achieve the desired order. The process is also numerically stable.

MATCONTROL NOTE: The routine **ordersch** in MATCONTROL can be used to order the eigenvalues in the real Schur form of the matrix.

FORTRAN ROUTINE: The Fortran routine STRSYL in LAPACK reorders the Schur decomposition of a matrix in order to find an orthonormal basis of a right invariant subspace corresponding to selected eigenvalues.

Invariant subspace sensitivity: Sep-Function

Let $Q^* A Q = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$ be the Schur decomposition of A . Let $Q = [Q_1, Q_2]$. Define

$$\text{sep}(T_{11}, T_{22}) = \min_{X \neq 0} \frac{\| T_{11}X - XT_{22} \|_F}{\| X \|_F}.$$

Then it can be shown (Golub and Van Loan (1996), p. 325) that the reciprocal of $\text{sep}(T_{11}, T_{22})$ is a good measure of the sensitivity of the invariant subspace spanned by the columns of Q .

4.4 Power Iteration and Inverse Iteration

In several practical applications, one needs to compute only a few eigenvalues (usually the few largest or smallest ones) and the associated eigenvectors. A classical method, called the **Power Iteration**, can be used for this purpose.

Power Iteration

Suppose that A is diagonalizable and let $X = (x_1, x_2, \dots, x_n)$ be such that $X^{-1}AX = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$. Assume that λ_1 is the dominant eigenvalue, that is, $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$. Let $v^{(0)}$ be a unit vector. Then the power iteration method then can be described as follows:

For $k = 1, 2, \dots$ do until convergence

$$\begin{aligned} w &= Av^{(k-1)} \\ v^{(k)} &= w / \|w\|_2 \\ \lambda^{(k)} &= (v^{(k)})^*Av^{(k)} \end{aligned}$$

End.

Convergence. It can be shown (Datta (1995)) that the power iteration converges to the dominant eigenvalue λ_1 and the eigenvector x_1 if $v^{(0)}$ has a component in the direction of x_1 .

The convergence, however, can be very slow if $|\lambda_1|$ and $|\lambda_2|$ are close.

Inverse Iteration

The rate of convergence can be improved by applying the power iteration to $(A - \sigma I)^{-1}$, where σ is a shift. The process is called **inverse iteration**. The inverse iteration is a commonly used procedure to compute a selected number of eigenvectors of a matrix.

Since A is initially reduced to a Hessenberg matrix H for the QR iteration process, then it is natural to take advantage of the structure of the Hessenberg matrix H in the process of inverse iteration. The **Hessenberg Inverse iteration** can then be stated as follows:

Step 1. Reduce the matrix A to an upper Hessenberg matrix $H : PAP^T = H$.

Step 2. Compute an eigenvalue λ , whose eigenvector x is sought, using the implicit QR iteration method described in the previous section.

Step 3. Choose a unit-length vector $y_0 \in \mathbb{C}^n$.

For $k = 1, 2, \dots$ do until convergence

$$\begin{aligned} \text{Solve for } z^{(k)} : (H - \lambda I)z^{(k)} &= y^{(k-1)} \\ \text{Compute } y^{(k)} &= z^{(k)} / \|z^{(k)}\| \end{aligned}$$

End

Step 4. Recover the eigenvector x of the matrix $A : x = P^T y^{(k)}$, where $y^{(k)}$ is an approximation of the eigenvector y obtained at the end of Step 3.

Note: If y is an eigenvector of H , then $x = P^T y$ is the corresponding eigenvector of A .

Convergence and Efficiency. The Hessenberg inverse iteration is very inexpensive. Once an eigenvalue is computed, the whole process requires only $O(n^2)$ flops. It typically requires only 1 to 2 iterations to obtain an approximate acceptable eigenvector.

4.5 Computing the Singular Value Decomposition

The following algorithm known as the **Golub-Kahan-Reinsch algorithm** is nowadays a standard computational algorithm for computing the singular value decomposition. The algorithm comes in two stages:

Stage I The $m \times n$ matrix $A(m \geq n)$ is transformed to an upper $m \times n$ bidiagonal matrix by orthogonal equivalence:

$$U_0^T A V_0 = \begin{pmatrix} B \\ 0 \end{pmatrix} \quad (4.5.1)$$

where B is the $n \times n$ upper bidiagonal matrix given by

$$B = \begin{pmatrix} \sigma_1 & * & & 0 \\ & \ddots & * & \\ & & \ddots & \ddots \\ 0 & & & \sigma_n \end{pmatrix}$$

Stage II The transformed bidiagonal matrix B is further reduced by orthogonal equivalence to a diagonal matrix Σ using the QR iteration method; that is, orthogonal matrices U_1 and V_1 are constructed such that

$$U_1^T B V_1 = \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n). \quad (4.5.2)$$

The matrix, Σ is the matrix of singular values. The singular vector matrices U and V are given by $U = U_0 U_1, V = V_0 V_1$.

We will briefly describe Stage I here. For a description of Stage II, see Golub and Van Loan (1996, pp. 452-457).

Reduction to Bidiagonal Form

We show how Householder matrices can be employed to construct U_0 and V_0 in stage I.

The matrices U_0 and V_0 are constructed as the product of Householder matrices as follows: $U_0 = U_1 U_2 \cdots U_n$, and $V_0 = V_1 V_2 \cdots V_{n-2}$. Let's illustrate construction of U_1, V_1 and U_2, V_2 , and their role in the bidiagonalization process with $m = 5$ and $n = 4$.

First, a Householder matrix U_1 is constructed such that

$$A^{(1)} = U_1 A = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix}$$

Next, a Householder matrix V_1 is constructed such that

$$A^{(2)} = A^{(1)}V_1 = \begin{pmatrix} * & * & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} = \left(\begin{array}{c|cccc} * & * & 0 & 0 & 0 \\ \hline 0 & 0 & & & \\ 0 & 0 & & & \\ 0 & 0 & & & \\ 0 & 0 & & & \end{array} \right) A'$$

The process is now repeated with $A^{(2)}$; that is, Householder matrices U_2 and V_2 are constructed so that

$$U_2 A^{(2)} V_2 = \begin{pmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{pmatrix}$$

Of course, in this step, we will work with the 4×3 matrix A' rather than the matrix $A^{(2)}$. Thus, first the orthogonal matrices U'_2 and V'_2 will be constructed such that

$$U'_2 A' V'_2 = \begin{pmatrix} * & * & 0 \\ 0 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix},$$

then U_2 and V_2 will be constructed from U'_2 and V'_2 in the usual way, that is, by embedding them in identity matrices of appropriate orders. The process is continued until the bidiagonal matrix B is obtained.

Example 4.5.1 Let

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}$$

Step 1.

$$U_1 = \begin{pmatrix} -0.1474 & -0.4423 & -0.8847 \\ -0.4423 & 0.8295 & -0.3410 \\ -0.8847 & -0.3410 & 0.3180 \end{pmatrix}$$

$$A^{(1)} = U_1 A = \begin{pmatrix} -6.7823 & -8.2567 & -9.7312 \\ 0 & 0.0461 & 0.0923 \\ 0 & -0.9077 & -1.8154 \end{pmatrix}$$

Step 2.

$$V_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.6470 & 0.7625 \\ 0 & -0.5571 & 0.6470 \end{pmatrix}$$

$$A^{(2)} = A^{(1)}V_1 = \begin{pmatrix} -6.7823 & 12.7620 & 0 \\ 0 & -1.0002 & 0.0245 \\ 0 & 1.9716 & -0.4824 \end{pmatrix}$$

Step 3.

$$U_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.0508 & 0.9987 \\ 0 & 0.9987 & 0.0508 \end{pmatrix}$$

$$B = U_2 A^{(2)} = U_2 A^{(1)} V_1 = U_2 U_1 A V_1 = \begin{pmatrix} -6.7823 & 12.7620 & 0 \\ 0 & -1.0081 & -1.8178 \\ 0 & 0 & 0 \end{pmatrix}$$

Note that from the above expression of B , it immediately follows that zero is a singular value of A .

Flop Count: The Householder bidiagonalization algorithm requires $4mn^2 - \frac{4n^3}{3}$ flops. Stage II, that is, the process of iterative reduction of the bidiagonal matrix to a diagonal matrix containing the singular values requires $30n$ flops and $2n$ square roots. The matrices U and V can be accumulated with $6mn$ and $6n^2$ flops, respectively.

Stability: The Golub-Kahan-Reinsch algorithm is **numerically stable**. It can be shown that the process will yield orthogonal matrices U and V and a diagonal matrix Σ such that $U^T A V = \Sigma + E$, where $\|E\|_2 \approx \mu \|A\|_2$.

4.6 The Generalized Real Schur Form: The QZ algorithm

In this section, we describe two canonical forms for a pair of matrices (A, B) : The **Hessenberg-triangular** and the **Generalized real Schur form**. The Generalized real Schur form displays the eigenvalues of the matrix pencil $A - \lambda B$, as the real Schur form does for the matrix A .

Given $n \times n$ matrices A , and B ; a scalar λ , and a nonzero vector x satisfying

$$Ax = \lambda Bx$$

are, respectively called an **eigenvalue** and **eigenvector** for the pencil $A - \lambda B$. The eigenvalue problem itself is called **generalized eigenvalue problem**. The eigenvalues and eigenvectors of the generalized eigenvalue problem are often called **generalized eigenvalues** and **generalized eigenvectors**. The matrix pencil $A - \lambda B$ is often conveniently denoted by the pair (A, B) .

The pair (A, B) is called **regular** if $\det(A - \lambda B)$ is not identically zero. Otherwise, it is **singular**. We will consider only regular pencil here. If B is nonsingular, then the eigenvalues of the regular pair (A, B) are finite and are the same as those of AB^{-1} or $B^{-1}A$.

If B is singular, and if the degree of $\det(A - \lambda B)$ is $r (< n)$, then $n - r$ eigenvalues of (A, B) are ∞ , and the remaining ones are the zeros of $\det(A - \lambda B)$.

As we will see later, the generalized real Schur form is an important tool in the numerical solutions of the **discrete algebraic Riccati equation** and the **Riccati equations with singular and ill-conditioned control weighting matrices** (Chapter 13).

The QZ algorithm

Assume that B is nonsingular. Then the basic idea is to apply the QR iteration algorithm to the matrix $C = B^{-1}A$ (or to AB^{-1}), without explicitly forming the matrix C . For if B is nearly singular, then it is not desirable to form B^{-1} . In this case the entries of C will be much larger than those of A and B , and the eigenvalues of C will be computed inaccurately. (Note that the eigenvalues of $B^{-1}A$ are the same as those of AB^{-1} , because $AB^{-1} = B(B^{-1}A)B^{-1}$). If AB^{-1} or $B^{-1}A$ is not to be computed explicitly, then the next best alternative, of course, is to transform A and B simultaneously to some reduced forms such as the triangular forms and then extract the generalized eigenvalues from these reduced forms. The simultaneous reduction of A and B to triangular forms by equivalence is guaranteed by the following theorem:

Theorem 4.6.1 (The Generalized Real Schur Decomposition) Given two $n \times n$ real matrices A and B , there exist orthogonal matrices Q and Z such that $Q^T A Z$ is an upper Real Schur matrix and $Q^T B Z$ is upper triangular:

$$\begin{aligned} Q^T A Z &\equiv A', \text{ an upper real Schur matrix.} \\ Q^T B Z &\equiv B', \text{ an upper triangular matrix.} \end{aligned}$$

The pair (A', B') is said to be in **generalized Real Schur Form**.

We shall now give a constructive proof of this theorem. The reduction to the generalized real Schur form is achieved in two stages.

Stage I. The matrices A and B are reduced to an upper Hessenberg and an upper triangular matrix, respectively, by simultaneous orthogonal equivalence:

$$\begin{aligned} A &\equiv Q^T A Z, \text{ an upper Hessenberg matrix} \\ B &\equiv Q^T B Z, \text{ an upper triangular matrix} \end{aligned}$$

Stage II. The Hessenberg-Triangular pair (A, B) is further reduced to the **generalized real Schur form** by applying **implicit QR iteration** to AB^{-1} .

This process is known as the **QZ Algorithm**.

We will now briefly sketch these two stages in the sequel.

4.6.1 Reduction to Hessenberg-Triangular Form

Let A and B be two $n \times n$ matrices. Then

Step 1. Find an orthogonal matrix U such that

$$B \equiv U^T B,$$

is an upper triangular matrix.

Form

$$A \equiv U^T A$$

(in general, A will be full).

Step 2. Reduce A to Hessenberg form while preserving the triangular structure of B .

The Step 2 is achieved as follows:

To start with, we have

$$\begin{aligned} A \equiv U^T A &= \begin{pmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & & & \\ * & * & \cdots & * \\ * & * & \cdots & * \end{pmatrix}, \\ B \equiv U^T B &= \begin{pmatrix} * & * & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ 0 & 0 & \ddots & \cdots & * \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots 0 & * \end{pmatrix}. \end{aligned}$$

First, the $(n, 1)^{th}$ entry of A is made zero by applying a Givens rotation $Q_{n-1,n}$ in the $(n-1, n)$ plane:

$$A \equiv Q_{n-1,n} A = \begin{pmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & & & \\ * & * & \cdots & * \\ 0 & * & \cdots & * \end{pmatrix}.$$

This transformation, when applied to B from the left, will give a fill-in in the $(n, n-1)$ position (denoted by t):

$$B \equiv Q_{n-1,n} B = \begin{pmatrix} * & * & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & t & * \end{pmatrix}.$$

The Givens rotation $Z_{n-1,n} = J(n-1, n, \theta)$ is now applied to the right of B to make the $(n, n-1)$ entry of B zero. Fortunately, this rotation, when applied to the right of A , does not destroy the zero produced earlier. Schematically, we have

$$B \equiv B Z_{n-1,n} = \begin{pmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & & \ddots & \ddots & \\ 0 & 0 & \cdots & 0 & * \end{pmatrix},$$

$$A \equiv AZ_{n-1,n} = \begin{pmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \end{pmatrix}.$$

The entries $(n-1, 1), (n-2, 1), \dots, (3, 1)$ of A are now successively made zero, each time applying an appropriate rotation to the left of A , followed by another appropriate Givens rotation to the right of B to zero out the undesirable fill-in in B . At the end of the first step, the matrix A is Hessenberg in its first column, while B remains upper triangular:

$$A = \begin{pmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{pmatrix}, \quad B = \begin{pmatrix} * & * & \cdots & \cdots & * \\ * & * & \cdots & \cdots & * \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & * \\ 0 & \cdots & 0 & * & * \end{pmatrix}.$$

The zeros are now produced on the second column of A in the appropriate places while retaining the triangular structure of B in an analogous manner.

The process is continued until the matrix A is an upper Hessenberg matrix while keeping B in upper triangular form.

Example 4.6.1

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 4 \\ 1 & 3 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{pmatrix}.$$

1. Form the Givens rotation Q_{23} to make a_{31} zero:

$$Q_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & .7071 & .7071 \\ 0 & -.7071 & .7071 \end{pmatrix},$$

$$A \equiv A^{(1)} = Q_{23}A = \begin{pmatrix} 1 & 2 & 3 \\ 1.4142 & 4.2426 & 4.9497 \\ 0 & 0 & -0.7071 \end{pmatrix}.$$

2. Update B :

$$B \equiv B^{(1)} = Q_{23}B = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0.7071 & 2.8284 \\ 0 & -0.7071 & 0 \end{pmatrix}.$$

3. Form the Givens rotation Z_{23} to make b_{32} zero:

$$Z_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix},$$

$$B \equiv B^{(1)}Z_{23} = Q_{23}BZ_{23} = \begin{pmatrix} 1 & 1 & -1 \\ 0 & 2.8284 & -0.7071 \\ 0 & 0 & 0.7071 \end{pmatrix}.$$

4. Update A :

$$A \equiv A^{(1)}Z_{23} = Q_{23}AZ_{23} = \begin{pmatrix} 1 & 3 & -2 \\ 1.4142 & 4.9497 & -4.2426 \\ 0 & -0.7071 & 0 \end{pmatrix}.$$

Now A is an upper Hessenberg and B is in upper triangular form.

4.6.2 Reduction to the Generalized Real Schur Form

At the beginning of this process, we have A and B as an upper Hessenberg and an upper triangular matrix, respectively, obtained from Stage 1. We can assume without loss of generality that the matrix A is an unreduced upper Hessenberg matrix. **The basic idea now is to apply an implicit QR step to AB^{-1} without ever forming this matrix explicitly.** We sketch just the basic ideas here. For details, see Datta (1995, pp. 500-504).

Thus a QZ step, analogous to an implicit QR step, will be as follows:

- Compute the first column n_1 of $N = (C - \alpha_1 I)(C - \alpha_2 I)$, where $C = AB^{-1}$, and α_1 and α_2 are suitably chosen shifts, without explicitly forming the matrix AB^{-1} .
(Note that n_1 has only three nonzero entries and the rest are zero)
- Find a Householder matrix Q_1 , such that $Q_1 n_1$ is a multiple of e_1 .
- Form $Q_1 A$ and $Q_1 B$.
- Simultaneously transform $Q_1 A$ to an upper Hessenberg matrix A_1 , and $Q_1 B$ to an upper triangular matrix B_1 : $A_1 \equiv Q'(Q_1 A)Z$: an upper Hessenberg;
 $B_1 \equiv Q'(Q_1 B)Z$: an upper triangular.

Using the implicit Q theorem (Theorem 4.2.1) we can show that the matrix $A_1 B_1^{-1}$ is essentially the same as that would have been obtained by applying an implicit QR step directly to AB^{-1} .

Application of a few QZ steps in sequence will then yield a quasi-triangular matrix $R = Q^T AZ$ and an upper triangular $T = Q^T BZ$, from which the generalized eigenvalues can be easily extracted.

Obtaining the Transforming Matrices

The transforming matrices Q and Z are obtained as follows:

The matrix $Q : Q = Q_1 Q_2 Q_3 \cdots Q_{n-1}$, The matrix $Z : Z = Z_1 Z_2 \cdots Z_{n-1}$. **Note that Q has the same first row as Q_1 .**

Choosing the Shifts

The double shifts α_1 and α_2 at a QZ step can be taken as the eigenvalues of the lower 2×2 submatrix of $C = AB^{-1}$. **The 2×2 lower submatrix of C again can be computed without explicitly forming B^{-1}** (see, Datta (1995), p. 501).

Algorithm 4.6.1 The Complete QZ Algorithm for Reduction to Generalized Schur Form

Inputs: Real $n \times n$ matrices A and B .

Outputs: The pair (R, T) of the generalized real Schur form of the pencil $A - \lambda B$. The matrix R is Quasi-triangular and T is upper triangular.

(I) Transform (A, B) to a Hessenberg-Triangular pair by orthogonal equivalence:

$$\begin{aligned} A &\equiv Q^T AZ, \text{ an upper Hessenberg,} \\ B &\equiv Q^T BZ, \text{ an upper Triangular.} \end{aligned}$$

(II) Apply a sequence of the QZ steps to the Hessenberg-Triangular pair (A, B) to produce $\{A_k\}$ and $\{B_k\}$, with properly chosen shifts.

(III) Monitor the convergence of the sequences $\{A_k\}$ and $\{B_k\}$:

$$\begin{aligned} \{A_k\} &\longrightarrow R, \text{ quasi-triangular (in real Schur form)} \\ \{B_k\} &\longrightarrow T, \text{ upper triangular.} \end{aligned}$$

Flop-Count: The implementation of I to III requires about $30n^3$ flops. The formation of Q and Z , if required, needs, respectively, another $16n^3$ and $20n^3$ flops (from experience it is known that about two QZ steps per eigenvalue are adequate).

Numerical Stability Properties: The QZ iteration algorithm is as **stable** as the QR iteration algorithm. It can be shown that the computed \hat{R} and \hat{S} satisfy

$$\begin{aligned} Q_0^T (A + E) Z_0 &= \hat{R} \\ Q_0^T (B + F) Z_0 &= \hat{S}, \end{aligned}$$

where Q_0 and Z_0 are orthogonal, $\|E\| \cong \mu \|A\|$, and $\|F\| \cong \mu \|B\|$, μ is the machine precision.

4.7 Computing of the Eigenvectors of the Pencil $A - \lambda B$.

Once an approximate generalized eigenvalue λ is computed, the corresponding eigenvector v of the pencil $A - \lambda B$ can be computed using the inverse iteration as before.

Step 1. Choose an initial eigenvector v_0 .

Step 2. For $k = 1, 2, \dots$ do until convergence

Solve $(A - \lambda B)\hat{v}_k = Bv_{k-1}$;

$$v_k = \hat{v}_k / \|\hat{v}_k\|_2.$$

A Remark on Solving $(A - \lambda B)\hat{v}_k = Bv_{k-1}$.

In solving $(A - \lambda B)\hat{v}_k = Bv_{k-1}$, substantial savings can be made by exploiting the Hessenberg-triangular structure to which the pair (A, B) is reduced as a part of the QZ algorithm. Note that in this case for a given λ , the matrix $A - \lambda B$ is also a Hessenberg matrix. Thus, at each iteration, only a Hessenberg system needs to be solved, which requires only $O(n^2)$ flops, compared to $O(n^3)$ flops required for a system with a full matrix.

Example 4.7.1

$$\begin{aligned} A &= 10^9 \begin{pmatrix} 3 & -1.5 & 0 \\ -1.5 & 3 & -1.5 \\ 0 & -1.5 & 1.5 \end{pmatrix}, \\ B &= 10^3 \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{pmatrix}. \end{aligned}$$

λ_1 = a generalized eigenvalue of $(A - \lambda B) = 1950800$.

$$v_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

k=1: Solve for v_1 :

Solve: $(A - \lambda_1 B)\hat{v}_1 = Bv_0$

$$\hat{v}_1 = \begin{pmatrix} .0170 \\ -0.0102 \\ 0.0024 \end{pmatrix}$$

$$v_1 = \hat{v}_1 / \|\hat{v}_1\| = \begin{pmatrix} .8507 \\ -.5114 \\ 0.1217 \end{pmatrix}.$$

MATLAB and MATCOM Notes.

The MATLAB function **qz** in the form: $[AA, BB, Q, Z, V] = \mathbf{qz}(A, B)$ produces upper triangular matrices AA and BB , and the orthogonal matrices Q and Z such that $QAZ = AA$, $QBZ = BB$.

The matrix V contains the eigenvectors. The generalized eigenvalues are obtained by taking the ratios of the corresponding diagonal entries of AA and BB .

The MATLAB function **eig** (A, B) gives only the generalized eigenvalues of the pencil $A - \lambda B$ from the Generalized Schur Decomposition.

MATCOM functions HESSTRI and INVITRGN compute, respectively, the Hessenberg-triangular reduction of the pair (A, B) and the eigenvectors of the pencil $A - \lambda B$ using inverse iteration.

Deflating Subspace for the Pencil $A - \lambda B$.

A k -dimensional subspace $S \in \mathbb{R}^n$ is a **deflating subspace** of the pencil $A - \lambda B$ if the subspace $\{Ax + By \mid x, y \in S\}$ has dimension k or less.

It can be easily seen that the **columns of Z in the generalized Schur decomposition form a family of deflating subspaces**. Also, $\text{span}\{Az_1, \dots, Az_k\}$ and $\text{span}\{Bz_1, \dots, Bz_k\}$ belong to $\text{span}\{q_1, \dots, q_k\}$, where z_i and q_i are, respectively, the columns of Z and Q .

Remark: In solving algebraic Riccati equations, deflating subspaces with specified spectrum need to be compute. There exist FORTRAN routines for computing such deflating subspaced developed by Van Dooren (1982).

4.8 Summary and Review

I. Numerical Instability in Obtaining Jordan and Companion Matrices

The Jordan canonical form and a companion form of a matrix, because of their rich structures, are important theoretical tools. Using these two decompositions, many important results in control theory have been established (see Kailath (1980)).

Unfortunately, however, these two forms cannot be obtained in a numerically stable way in general. Since it is necessary to use non-orthogonal transformations to achieve these forms, the transforming matrices can be highly ill-conditioned. Some discussions to this effect have been given in **Section 4.1**. Because of possible numerical instabilities in reduction of A to a companion matrix, and the fact that the zeros of a polynomial can be extremely sensitive to small perturbations, **it is not advisable to compute the eigenvalues of a matrix by finding the zeros of its characteristic polynomial**.

II. Hessenberg and Real Schur Forms

Both Hessenberg and real Schur forms can be obtained via orthogonal similarity transformations. These two forms, thus, are extremely valuable tools in numerical computations. In fact, many of the numerically effective algorithms for control problems described in this book, are based on these two forms.

Reduction to Hessenberg Form

A Hessenberg form, via orthogonal similarity transformation, is obtained using either Householder or Givens transformations. The Householder method for Hessenberg reduction is described in Section 4.2. For a description of Givens Hessenberg reduction, see Datta (1995). The **implicit Q theorem (Theorem 4.2.1)** guarantees that the two Hessenberg forms obtained by Householder's and Givens' methods are essentially the same.

Real Schur Form: Computing the Eigenvalues, Eigenvectors, and Orthonormal Bases for Invariant Subspaces

The real Schur form (RSF) of a matrix is a **quasi-triangular matrix** whose diagonal entries are either scalars or 2×2 matrices.

Every real matrix A can be transformed to RSF by an orthogonal similarity.

Since the RSF of a matrix A displays the eigenvalues of A , any numerical method for obtaining the RSF of order higher than four X has to be iterative in nature. The standard method for obtaining the RSF is the QR iteration method with implicit double shift. This method is described in some details in Sections 4.3.1-4.3.4. **The double shift implicit QR iteration method is nowadays the standard method for finding the eigenvalues of a matrix.**

An orthonormal basis for the invariant subspace associated with a given set of eigenvalues can also be found by reordering the eigenvalues in RSF in a suitable way. This is discussed in **Section 4.3.5**.

Once the RSF is found, it can be employed to compute the eigenvectors of A . This is not discussed here. The interested readers are referred to Datta (1995, pp. 452-455). Instead, a commonly used procedure for computing selected eigenvectors , called the **inverse iteration method, is described in Section 4.4**. A brief description of the well-known **power method** to compute the dominant eigenvalue and the associated eigenvector also given in this section.

III. Computing the SVD of a Matrix

The standard method for computing the SVD, called the **Golub-Kahan-Reinsch algorithm**, is described in Section 4.5. The method comes in two stages:

Stage I. Reduction of the matrix A to a bidiagonal form.

Stage II. Further reduction of the bidiagonal matrix obtained in Stage I to a diagonal matrix using implicit QR iteration.

The detailed discussion of Stage II is omitted here. The readers are referred to Golub and Van Loan (1996, pp. 452-456).

IV. The Generalized Real Schur Form.

The generalized real Schur form of a pair of matrices (A, B) is a matrix-pair (A', B') , where A' is an upper real Schur matrix and B' is an upper triangular matrix (**Theorem 4.6.1**).

The standard method for computing the general real Schur form is the **QZ iteration algorithm**. The QZ algorithm also comes in two stages:

Stage I. Reduction of (A, B) to Hessenberg-triangular form.

Stage II. Further reduction of the Hessenberg-triangular form obtained in Stage I to the generalized real Schur form.

Stage I is a finite procedure. Again, the Householder or Givens transformations can be used.

The Householder procedure is described in **Section 4.6.1**.

Stage II is an iterative procedure. Only a brief sketch of the procedure is presented here in **Section 4.6.2**. For details, readers are referred to Datta (1995, pp. 500-504).

The generalized real Schur form displays the eigenvalues (called generalized eigenvalues) of the linear pencil $A - \lambda B$. Once the eigenvalues are obtained, the selected eigenvectors can be computed using **generalized inverse iteration** (**Section 4.7**).

4.9 Chapter Notes and Further Reading

The material of this chapter has been taken from the recent book of the author (Datta (1995)). For advanced readings of the topics dealt with in this chapter, consult the book by Golub and Van Loan (1996). For a description of the toolbox *MATCOM* and how to obtain it, see the section on Chapter Notes and Further Reading of Chapter 3 (**Section 3.11**).

References

1. B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, CA, Pacific Grove, 1995.
2. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Edition, The Johns Hopkins University Press, Baltimore, MD, 1996.
3. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Otrouchov, and D. Sorensen, *LAPACK Users' Guide*, 2nd Edition, SIAM, Philadelphia, 1995.
4. MATLAB *User's Guide*, The Math Works, Inc., Natick, MA, 1992.
5. T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.
6. G.W. Stewart, Algorithm 406. HWR3 and EXCHNG: FORTRAN programs for calculating the eigenvalues of a real upper Hessenberg matrix in a prescribed order, *ACM Trans. Math. Soft.* vol.2, pp. 275-280, 1976.

7. P. Van Dooren, Algorithm 590—DSUBSP and EXCHQZ: Fortran Subroutines for Computing Deflating Subspaces With Specified Spectrum, *ACM Trans. Math. Software*, 8, 376–382 (1982).
8. J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, England, 1965.

Part II

Control Systems Analysis

- Chapter 5.** Linear State Space Models and Solutions of the State Equations
- Chapter 6.** Controllability, Observability and Distance to Uncontrollability
- Chapter 7.** Stability, Inertia, and Robust Stability
- Chapter 8.** Numerical Solutions and Conditioning of Lyapunov and Sylvester Equations

Chapter 5

LINEAR STATE SPACE MODELS AND SOLUTIONS OF THE STATE EQUATIONS

Contents

5.1	Introduction	136
5.2	State-Space Representations of Control Systems	137
5.2.1	Continuous-Time Systems	137
5.2.2	Discrete-Time Systems	148
5.2.3	Descriptor Systems	149
5.3	Solutions of a Continuous-Time System: System Responses	152
5.3.1	Some Important Properties of the Matrix e^{At}	156
5.3.2	Sensitivity of e^{At}	157
5.3.3	Computational Methods for e^{At}	158
5.3.4	Comparison of Different Methods for Computing the Exponential Matrix	166
5.3.5	Evaluating an Integral with the Exponential Matrix	167
5.4	State-Space Solution of the Discrete-Time System	168
5.5	Transfer Function and Frequency Response	170
5.5.1	Transfer Function	170
5.5.2	The Frequency Response Matrix and its Computation	172
5.6	Some Selected Software	176
5.6.1	MATLAB CONTROL SYSTEM TOOLBOX	176
5.6.2	MATCONTROL	176
5.6.3	SLICOT	176
5.6.4	MATRIX _X	177
5.7	Summary and Review	178

Topics Covered

- State-Space Models
- Solutions of the State Equations
- System Responses
- Sensitivity Analysis of the Matrix Exponential Problem
- Numerical Methods for Computing the Exponential Matrix and the Integral involving an Exponential Matrix
- Computation of the Frequency Response Matrix

5.1 Introduction

A finite-dimensional time-invariant linear continuous-time dynamical system may be described using the following system of first-order ordinary differential equations:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t).\end{aligned}$$

The input and the output of the system are defined in continuous-time over the interval $[0, \infty)$. The system is, therefore, known as the **continuous-time system**. The discrete-time analog of this system is the system of difference equations:

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k).\end{aligned}$$

We will consider in this book only time-invariant systems;
that is, the matrices A, B, C and D will be assumed constant matrices throughout the book.

It is first shown in Section 5.2 how some simple familiar physical systems can be described in state-space forms. Very often the mathematical model of a system is not obtained in first-order form; it may be a system of nonlinear equations, a system of **second-order differential equations or partial differential equations**. It is shown how such systems can be reduced to the standard first-order state-space forms.

The computational methods for the state equations are then considered both in time and frequency-domain.

The major component of the time-domain solution of a continuous-time system is the exponential matrix e^{At} . Some results on the sensitivity of this matrix and various well-known methods for its computation: the Taylor series method, the Padé approximation method, the methods based on decompositions of A , the ordinary-differential equation methods, etc. are described in Section 5.3. A comparative study of these methods is also included. **The Padé method (Algorithm 5.3.1) (with scaling and squaring)** and the **method, based on the Real Schur decomposition of A (Algorithm 5.3.2)**, are recommended for practical use. This section concludes with an algorithm for numerically computing an integral with an exponential matrix (**Algorithm 5.3.3**).

Section 5.4 describes the state-space solution of a **discrete-time system**. The major computational task here is computation of various powers of A .

In **Section 5.5** the problem of computing the **frequency response matrix** for many different values of the frequencies is considered. The computation of the frequency response matrix is necessary to study various system responses in frequency domain. A widely-used method (**Algorithm 5.5.1**), based on the one-time reduction of the state matrix A to a Hessenberg matrix, is described in detail and the references to the other recent methods are given.

Reader's Guide for Chapter 5.

The readers familiar with basic concepts of modeling and state-space systems can skip Sections 5.2, 5.4, and 5.5.1.

5.2 State-Space Representations of Control Systems

5.2.1 Continuous-Time Systems

Consider the dynamical system represented by means of the following system of ordinary first-order differential equations:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0, \quad (5.2.1)$$

$$y(t) = Cx(t) + Du(t) \quad (5.2.2)$$

In this description,

$x(t)$ is an n -dimensional vector, called system **state**

$u(t)$ is an m -dimensional vector ($m \leq n$), called system **input**

$y(t)$ is an r -dimensional vector, called system **output**.

The vector $x(t_0)$ is the **initial condition** of the system. The components of $x(t)$ are called **state variables**.

The matrices A, B, C, and D are **time-invariant matrices**, respectively, of dimensions $n \times n$, $n \times m$, $r \times n$, and $r \times m$. The above representation is known as a time-invariant **continuous-time state-space model** of a dynamical system.

Schematically, the model is represented in Figure 5.1.

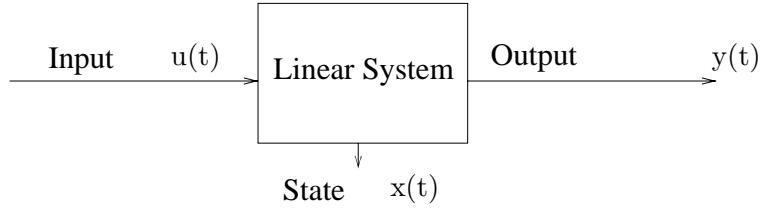


Figure 5.1. Representation of a Continuous-time State-Space Model.

Clearly, at a given time t , the variables arriving at the system would form the input, those internal to the system form the state, while the others that can be measured directly comprise the output.

The space $X \subseteq \mathbb{R}^n$, where all the states lie for all $t \geq 0$ is called the **state-space**, the equation (5.2.1) is called the **state equation** and the equation (5.2.2) is called the **output equation**. If $m = r = 1$, the system is said to be **single-input single-output (SISO) system**. *The system represented by the equations (5.2.1) and (5.2.2) is sometimes written compactly as (A, B, C, D) or as (A, B, C) , in case D is not used in modeling. Sometimes $\dot{x}(t)$ and $x(t)$ will be written just by \dot{x} and x , for the sake of convenience.* Similarly, $u(t)$ and $y(t)$ will be written as u and y , respectively.

We provide below a few examples to illustrate the state-space representations of some simple systems.

Example 5.2.1 A parallel RLC circuit

Consider the following parallel RLC circuit excited by the current source $u(t)$ and with output $y(t)$.

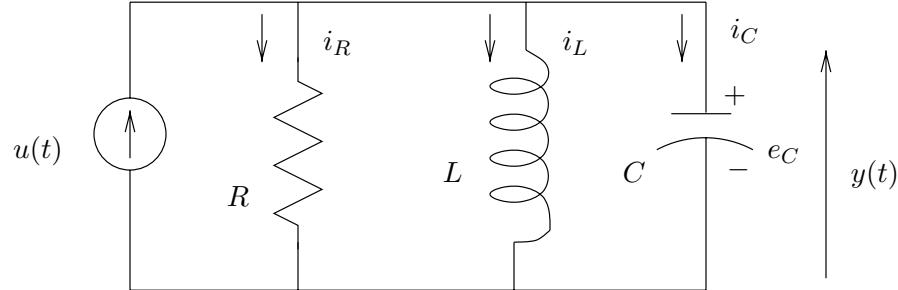


Figure 5.2. A Parallel RLC Circuit.

The current and voltage equations governing the circuit are:

$$u = i_R + i_L + i_C; \quad i_C = C \frac{de_C}{dt}; \quad e_C = L \frac{di_L}{dt} = Ri_R$$

Defining the states by $x_1 := i_L$ and $x_2 := e_C$, the state and output equations are, respectively:

$$\dot{x} = Ax + bu \text{ and } y = cx$$

where $x = [x_1, x_2]^T$,

$$A = \begin{bmatrix} 0 & \frac{1}{L} \\ -\frac{1}{C} & -\frac{1}{RC} \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \frac{1}{C} \end{bmatrix}, \quad c = [0 \ 1]$$

Example 5.2.2 Consider again another electric network, as shown in the following diagram:

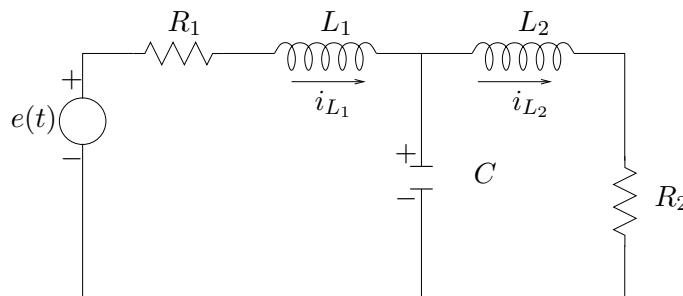


Figure 5.3. An Expanded RLC Circuit

The state variables here are taken as voltage across the capacitor and the current through the inductor. The state equations are

$$\begin{aligned} L_1 \frac{di_{L_1}(t)}{dt} &= -R_1 i_{L_1}(t) - e_C(t) + e(t) \\ L_2 \frac{di_{L_2}(t)}{dt} &= -R_2 i_{L_2}(t) + e_C(t) \\ C \frac{de_C(t)}{dt} &= i_{L_1}(t) - i_{L_2}(t) \end{aligned}$$

Setting $x_1 = i_{L_1}$, $x_2 = i_{L_2}$, $x_3 = e_C$, $b = \begin{pmatrix} 1 \\ \frac{1}{L_1} \\ 0 \\ 0 \end{pmatrix}$, $u = e(t)$, the matrix form of the state-space representation of the above system is given by

$$\dot{x}(t) = \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \begin{pmatrix} -\frac{R_1}{L_1} & 0 & -\frac{1}{L_1} \\ 0 & -\frac{R_2}{L_2} & \frac{1}{L_2} \\ \frac{1}{C} & -\frac{1}{C} & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} + bu(t).$$

State-Space Representation of Second-order Systems

Mathematical models of several practical problems, especially those arising in vibrations of structures, are second-order differential equations.

We show by means of a simple example of a spring-mass system how the equations of motion represented by a second-order differential equation can be converted to a first-order state-space representation.

Example 5.2.3 A Spring-Mass System

Consider the following spring-mass system with equal spring constants k and masses m_1 and m_2 . Let the force u_1 be applied to the mass m_1 and u_2 be applied to the mass m_2 .

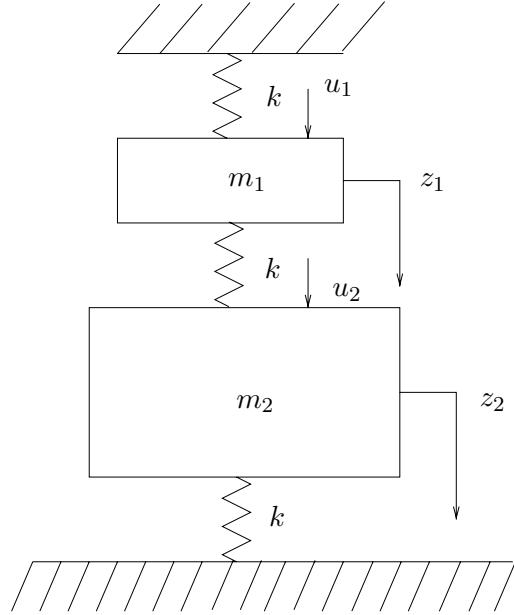


Figure 5.4. A Spring-Mass System

The equations of motion for the system are

$$\begin{aligned} m_1 \ddot{z}_1 + k(z_1 - z_2) + kz_1 &= u_1 \\ m_2 \ddot{z}_2 - k(z_1 - z_2) + kz_2 &= u_2 \end{aligned} \quad (5.2.3)$$

or in matrix-form:

$$\begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix} \begin{pmatrix} \ddot{z}_1 \\ \ddot{z}_2 \end{pmatrix} + \begin{pmatrix} 2k & -k \\ -k & 2k \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \quad (5.2.4)$$

Set

$$z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

then we have

$$M\ddot{z} + Kz = u, \quad (5.2.5)$$

where

$$M = \text{diag}(m_1, m_2), \quad K = \begin{pmatrix} 2k & -k \\ -k & 2k \end{pmatrix} \text{ and } u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

Let us make a change of variables from z to x as follows:

Set

$$x_1 = z \text{ and } x_2 = \dot{z}.$$

Then, in terms of the new variables, the equations of motion become

$$\begin{aligned}\dot{x}_1 &= x_2 \\ M\dot{x}_2 &= -Kx_1 + u\end{aligned}\tag{5.2.6}$$

or

$$\dot{x} = \begin{pmatrix} 0 & I \\ -M^{-1}K & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ M^{-1} \end{pmatrix} u\tag{5.2.7}$$

where

$$x = (x_1, x_2)^T = (z, \dot{z})^T.$$

The equation (5.2.7) is a first-order representation of the second-order system (5.2.4).

State-space Representations of Nonlinear Systems.

Mathematical models of many real-life applications are nonlinear systems of differential equations.

Very often it is possible to linearize a nonlinear system, and then after linearization, the first-order state-space representation of transformed linear system can be obtained.

We will illustrate this by means of the following well-known examples (see Luenberger (1979), Chen (1984), Szidarovszky and Bahill (1991), etc.).

Example 5.2.4 Balancing a stick

Consider the simple problem of balancing a stick on your hand as shown in the following figure:

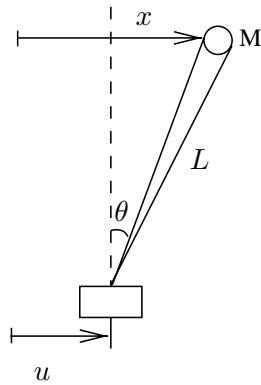


Figure 5.5. Balancing of a Stick

Here L is the length of the stick, and M is the mass of the stick concentrated on the top. The input $u(t)$ is the position of the hand. Then the position of the top of the stick is

$$x(t) = L \sin \theta(t) + u(t).$$

The torque due to gravity acting on the mass is $MgL \sin \theta(t)$. The rotational inertial of the mass on the stick is $ML^2 \ddot{\theta}(t)$. **The shift of the inertial term down to the pivot point is $\ddot{u}(t)ML \cos \theta(t)$.**

Thus we have

$$MgL \sin \theta(t) = ML^2 \ddot{\theta}(t) + \ddot{u}(t)ML \cos \theta(t).$$

The above equations are clearly nonlinear. We now linearize these equations by assuming that θ is small. We then can take $\cos \theta = 1$, $\sin \theta = \theta$.

This gives us

$$x(t) = L\theta(t) + u(t)$$

and

$$MgL\theta(t) = ML^2 \ddot{\theta}(t) + \ddot{u}(t)ML.$$

Eliminating $\theta(t)$ from these two equations, we obtain

$$\ddot{x}(t) = \frac{g}{L} (x(t) - u(t)).$$

We can now write down the first-order state-space representation by setting $v(t) = \dot{x}(t)$.

The first-order system then is:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{v}(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{g}{L} & 0 \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} + \begin{pmatrix} 0 \\ -\frac{g}{L} \end{pmatrix} u(t).$$

Example 5.2.5 A Cart with an Inverted Pendulum

Next we consider a similar problem, but this time with some more forces exerted. (Taken from Chen (1984), 96-98).

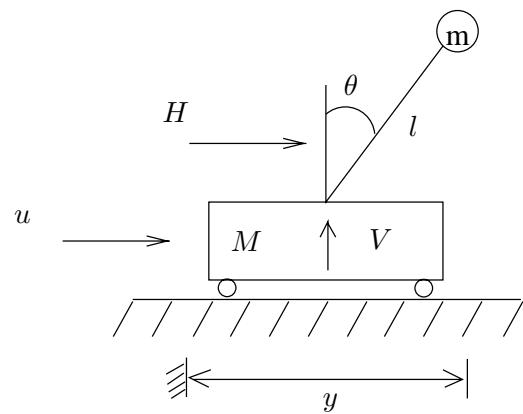


Figure 5.6. A Cart with an Inverted Pendulum

In the figure above, a cart is carrying an inverted pendulum with mass m and length l . Let M be the mass of the cart. Let H and V be, respectively, the horizontal and vertical forces exerted by the cart to the pendulum. Newton's law applied to the linear movements gives

$$\begin{aligned} M\ddot{y}(t) &= u - H \\ H &= m\ddot{y} + ml \cos \theta \ddot{\theta} - ml \sin \theta (\dot{\theta})^2 \\ \text{and } mg - V &= ml \left(-\sin \theta \ddot{\theta} - \cos \theta (\dot{\theta})^2 \right) \end{aligned}$$

Newton's law applied to the rotational movement of the pendulum gives

$$ml^2\ddot{\theta} = mgl \sin \theta + Vl \sin \theta - Hl \cos \theta$$

These are nonlinear equations. We now linearize them by making the same assumptions as before; that is, we assume that θ is small so that we can take $\sin \theta = \theta$, $\cos \theta = 1$. Dropping the terms involving θ^2 , $\dot{\theta}^2$, $\theta\dot{\theta}$, and $\theta\ddot{\theta}$, and setting $\sin \theta = \theta$, and $\cos \theta = 1$, we obtain, from above, by eliminating V and H

$$(M+m)\ddot{y} + ml\ddot{\theta} = u$$

and

$$2l\ddot{\theta} - 2g\theta + \ddot{y} = 0$$

Solving for \ddot{y} and $\ddot{\theta}$, we obtain

$$\begin{aligned} \ddot{y} &= -\frac{2gm}{2M+m}\theta + \frac{2}{2M+m}u \\ \ddot{\theta} &= \frac{2g(M+m)\theta}{(2M+m)l} - \frac{1}{(2M+m)l}u \end{aligned}$$

The state-space representations of these linear equations can now be written down by setting $x_1 = y$, $x_2 = \dot{y}$, $x_3 = \theta$, and $x_4 = \dot{\theta}$, as follows:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-2gm}{2M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{2g(M+m)}{(2M+m)l} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{2}{2M+m} \\ 0 \\ -\frac{1}{(2M+m)l} \end{pmatrix} u$$

$$y = (1, 0, 0, 0)x.$$

The nonlinear equations in Examples 5.2.4 and 5.2.5 are special cases of the general nonlinear equations of the form

$$\begin{aligned} \dot{\tilde{x}}(t) &= f(\tilde{x}(t), \tilde{u}(t), t), \quad \tilde{x}(t_0) = \tilde{x}_0 \\ \tilde{y}(t) &= h(\tilde{x}(t), \tilde{u}(t)), t, \end{aligned}$$

where f and h are vector functions. We will now show how these equations can be written in the standard first-order state-space form (Sayed (1994)).

Assume that the nonlinear differential equation

$$\dot{\tilde{x}}(t) = f(\tilde{x}(t), \tilde{u}(t), t), \quad \tilde{x}(t_0) = x_0$$

has a unique solution and this unique solution is also continuous with respect to the initial condition.

Let $\tilde{x}_{nom}(t)$ denote the unique solution corresponding to the given input $\tilde{u}_{nom}(t)$ and the given initial condition $\tilde{x}_{nom}(t_0)$.

Let the nominal data $\{\tilde{u}_{nom}(t), \tilde{x}_{nom}(t)\}$ be perturbed so that

$$\tilde{u}(t) = \tilde{u}_{nom}(t) + u(t)$$

and $\tilde{x}(t_0) = \tilde{x}_{nom}(t_0) + x(t_0)$, where $\|u(t)\|$ and $\|x(t_0)\|$ are small; $\|u(t)\| = \sup_t \|u(t)\|_2$.

Assume further that

$$\tilde{x}(t) = \tilde{x}_{nom}(t) + x(t),$$

and

$$\tilde{y}(t) = \tilde{y}_{nom}(t) + y(t)$$

where $\|x\|$ and $\|y\|$ are small.

These nonlinear equations can then be linearized (assuming that f and h are smooth enough) by expanding f and h around $(\tilde{u}_{nom}(t), \tilde{x}_{nom}(t))$, giving rise to a time-invariant linear state-space model of the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad x(t_0) = x_0 \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

where

$$\begin{aligned} A &= \left. \frac{\partial f}{\partial \tilde{x}} \right|_{(\tilde{x}_{nom}(t), \tilde{u}_{nom}(t))} & B &= \left. \frac{\partial f}{\partial \tilde{u}} \right|_{(\tilde{x}_{nom}(t), \tilde{u}_{nom}(t))} \\ C &= \left. \frac{\partial h}{\partial \tilde{x}} \right|_{(\tilde{x}_{nom}(t), \tilde{u}_{nom}(t))}, & D &= \left. \frac{\partial h}{\partial \tilde{u}} \right|_{(\tilde{x}_{nom}(t), \tilde{u}_{nom}(t))} \end{aligned}$$

Example 5.2.6 The Motion of a Satellite.

Suppose that a satellite of unit mass orbits the earth at a distance $d(t)$ from its center. Let $\theta(t)$ be the angular position of the satellite at time t , and the three forces acting on the satellite are: a radial force $u_1(t)$, a tangential force $u_2(t)$ and an attraction force $\alpha/d^2(t)$, where α is a constant.

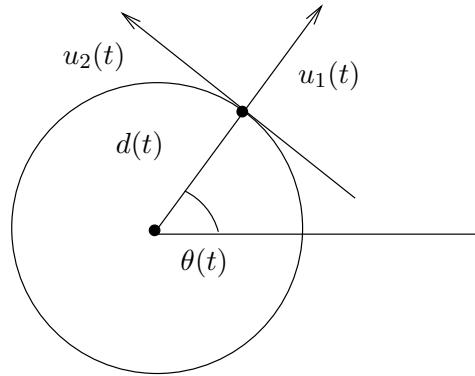


Figure 5.7. The Motion of a Satellite

The equations of motion are given by

$$\begin{aligned}\ddot{d}(t) &= d(t)\dot{\theta}^2(t) - \frac{\alpha}{d^2(t)} + u_1(t) \\ \ddot{\theta}(t) &= \frac{-2\dot{d}(t)\dot{\theta}(t)}{d(t)} + \frac{u_2(t)}{d(t)}.\end{aligned}$$

Let's define the state variable as

$$\tilde{x}_1(t) = d(t), \quad \tilde{x}_2(t) = \dot{d}(t), \quad \tilde{x}_3(t) = \theta(t), \quad \tilde{x}_4(t) = \dot{\theta}(t)$$

and the output variables as

$$\tilde{y}_1(t) = d(t), \quad \tilde{y}_2(t) = \theta(t).$$

The state-space model is then given by

$$\begin{aligned}\begin{pmatrix} \dot{\tilde{x}}_1(t) \\ \dot{\tilde{x}}_2(t) \\ \dot{\tilde{x}}_3(t) \\ \dot{\tilde{x}}_4(t) \end{pmatrix} &= \begin{pmatrix} \tilde{x}_2(t) \\ \tilde{x}_1(t)\tilde{x}_4^2(t) - \frac{\alpha}{\tilde{x}_1^2(t)} + u_1(t) \\ \tilde{x}_4(t) \\ \frac{-2\tilde{x}_2(t)\tilde{x}_4(t)}{\tilde{x}_1(t)} + \frac{u_2(t)}{\tilde{x}_1(t)} \end{pmatrix}, \\ \begin{pmatrix} \tilde{y}_1(t) \\ \tilde{y}_2(t) \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{x}_1(t) \\ \tilde{x}_2(t) \\ \tilde{x}_3(t) \\ \tilde{x}_4(t) \end{pmatrix},\end{aligned}$$

and the initial conditions are

$$\tilde{x}_0 = \tilde{x}(0) = \begin{pmatrix} \tilde{x}_1(0) \\ \tilde{x}_2(0) \\ \tilde{x}_3(0) \\ \tilde{x}_4(0) \end{pmatrix} = \begin{pmatrix} d(0) \\ 0 \\ \theta(0) \\ \omega_0 \end{pmatrix} = \begin{pmatrix} d_0 \\ 0 \\ \theta_0 \\ \omega_0 \end{pmatrix}$$

The above is still a nonlinear model of the form

$$\begin{aligned}\dot{\tilde{x}}(t) &= f(\tilde{x}(t), \tilde{u}(t), t), \quad \tilde{x}(t_0) = \tilde{x}_0 \\ \tilde{y}(t) &= h(\tilde{x}(t), \tilde{u}(t), t).\end{aligned}$$

Linearizing this nonlinear model around the initial point $(\tilde{x}(0), \tilde{u}(0))$, where $\tilde{u}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, we obtain the linear model:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t),\end{aligned}$$

where

$$\begin{aligned}A &= \left. \frac{\partial f}{\partial \tilde{x}} \right|_{(\tilde{x}(0), \tilde{u}(0))} \\ &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 3\omega_0^2 & 0 & 0 & 2d_0\omega_0 \\ 0 & 0 & 0 & 1 \\ 0 & -2\frac{\omega_0}{d_0} & 0 & 0 \end{pmatrix}, \\ B &= \left. \frac{\partial f}{\partial \tilde{u}} \right|_{(\tilde{x}(0), \tilde{u}(0))} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & \frac{1}{d_0} \end{pmatrix},\end{aligned}$$

and

$$C = \left. \frac{\partial h}{\partial \tilde{x}} \right|_{(\tilde{x}(0), \tilde{u}(0))} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

State-space Representation of Systems Modeled by Partial Differential Equations

Mathematical models of many engineering problems such as those arising in fluid dynamics, mechanical systems, heat transfer, etc. are partial differential equations. The discretizations of these equations naturally lead to state-space models. We illustrate the idea by means of the following example.

Example 5.2.7 Consider the partial differential equation

$$\frac{\partial^4 y}{\partial x^4} + \frac{P}{EI} \frac{\partial^2 y}{\partial t^2} = \frac{1}{EI} F(x, t),$$

which models the **deflection of a prismatic beam**. (Soong (1990), pp. 180-181).

Let $y(x, t)$ be the transverse displacement of a typical segment of the beam that is located at a distance x from the end, and $F(x, t)$ be the applied force. EI is the flexural rigidity, and P is the density of the material of the beam per unit length. Let L be the length of the beam.

Assume that the solution $y(x, t)$ can be written as

$$y(x, t) = \sum_{j=1}^n v_j(x) p_j(t)$$

($n = \infty$ in theory, but in practice it is large but finite). Also assume that

$$F(x, t) = \sum_{j=1}^r \delta(x - a_j) u_j(t),$$

where $\delta(\cdot)$ is the Dirac delta function.

That is, we assume that the force is point-wise and is exerted at r points of the beam.

Substituting these expressions of $y(x, t)$ and $F(x, t)$ in the partial differential equation, we have

$$\dot{z}(t) = \mathbf{A} z(t) + Bu(t),$$

where $z(t) = (p_1, \dot{p}_1, p_2, \dot{p}_2, \dots, p_n, \dot{p}_n)^T$,

$$\begin{aligned} \mathbf{A} &= \text{diag}(\Lambda_1, \Lambda_2, \dots, \Lambda_n), \quad B = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix}, \\ \Lambda_j &= \begin{pmatrix} 0 & 1 \\ -w_j^2 & 0 \end{pmatrix}, \quad w_j = \frac{j^4 \pi^4 EI}{b^4 L}. \end{aligned}$$

The matrices B_j and the vector u are defined by:

$$B_j = \frac{1}{EI} \begin{pmatrix} 0 & 0 & \cdots & 0 \\ v_j(a_1) & v_j(a_2) & \cdots & v_j(a_r) \end{pmatrix}_{n \times r} \quad \text{and} \quad u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{pmatrix}.$$

5.2.2 Discrete-Time Systems

A linear time-invariant discrete-time system may be represented by means of a system of difference equations:

$$x(k+1) = Ax(k) + Bu(k) \quad (5.2.8)$$

$$y(k) = Cx(k) + Du(k). \quad (5.2.9)$$

As before, $x(k)$ is the n -dimensional **state vector**, $u(k)$ is the m -dimensional **input vector**; and, A , B , C and D are time-invariant matrices of dimensions $n \times n$, $n \times m$, $r \times n$ and $r \times m$, respectively.

Sometimes we will write the above equations in the form:

$$x_{k+1} = Ax_k + Bu_k \quad (5.2.10)$$

$$y_k = Cx_k + Du_k. \quad (5.2.11)$$

Example 5.2.8 (Cohort Population Model)

Suppose that the population of a country is divided into a cohort of n age groups.

Let $p_i(t)$ denote the population of group i at time t . Let α_i denote the survival rate of group i . Let β_i denote the birth rate of group i . Then the population model can be represented by the following discrete-time system:

$$\begin{aligned} p_{i+1}(t+1) &= \alpha_i p_i(t), i = 1, \dots, n-1. \\ p_1(t+1) &= \beta_1 p_1(t) + \beta_2 p_2(t) + \dots + \beta_n p_n(t). \end{aligned}$$

Let $u_i(t)$ denote the number of people joining the group i at time t from the outside. Then the matrix-representation of the above model is

$$p(t+1) = Ap(t) + Iu(t),$$

where

$$A = \begin{pmatrix} \beta_1 & \beta_2 & \cdots & \cdots & \beta_n \\ \alpha_1 & 0 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & \alpha_{n-1} & 0 \end{pmatrix}, \quad u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}, \quad p = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}.$$

The output can be chosen as the total population P . Thus the output equation can be written as

$$\begin{aligned} P &= p_1(t) + p_2(t) + \dots + p_n(t) \\ &= (1, 1, \dots, 1)p(t). \end{aligned}$$

The matrix A is known as **Leslie matrix** after the name of its discoverer P.H. Leslie. See Luenberger (1979) for details.

5.2.3 Descriptor Systems

The system represented by equations (5.2.1) and (5.2.2) is a special case of a more general system, known as the **descriptor system**.

A continuous-time linear descriptor system has the form:

$$E\dot{x}(t) = Ax(t) + Bu(t) \quad (5.2.12)$$

$$y(t) = Cx(t) + Du(t) \quad (5.2.13)$$

Similarly, a discrete-time linear descriptor system has the form:

$$Ex(k+1) = Ax(k) + Bu(k) \quad (5.2.14)$$

$$y(k) = Cx(k) + Du(k) \quad (5.2.15)$$

If the matrix E is nonsingular, then, of course, the descriptor system represented by (5.2.12) and (5.2.13) is reduced to the standard form (5.2.1)-(5.2.2). Similarly, for the system (5.2.14)-(5.2.15).

However, the case when E is singular is more interesting.

We will now give an example to show how a singular descriptor systems arises in practice.

Example 5.2.9 (*Simplified Samuelson's Model of Economics*)

Let $NI(k)$, $CS(k)$, $IV(k)$, and $GE(k)$, denote, respectively, the national income, consumption, investment, and government expenditure of a country at a given year k .

The economist P.A. Samuelson proposed a model of the national economy of a country, which is based on the following assumptions:

1. National income $NI(k)$ is equal to the sum of the consumption $CS(k)$, investment $IV(k)$, and the government expenditure $GE(k)$ at a given year k .
2. Consumption $CS(k+1)$ at the year $k+1$ is proportional to the national income $NI(k)$ at the previous year k .
3. Investment $IV(k+1)$ at the year $k+1$ is proportional to the difference of the consumer spending $CS(k+1)$ at that year and that of the previous year $CS(k)$.

The state-space representation of Samuelson's model, then, can be written in the form:

$$\begin{aligned} NI(k) &= CS(k) + IV(k) + GE(k) \\ CS(k+1) &= \alpha NI(k) \\ IV(k+1) &= \beta [CS(k+1) - CS(k)] \end{aligned} \quad (5.2.16)$$

These equations in matrix form are:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & -\beta & 0 \end{pmatrix} \begin{pmatrix} IV(k+1) \\ CS(k+1) \\ NI(k+1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & -1 \\ 0 & 0 & \alpha \\ 0 & -\beta & 0 \end{pmatrix} \begin{pmatrix} IV(k) \\ CS(k) \\ NI(k) \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} GE(k).$$

or

$$Ex(k+1) = Ax(k) + Bu(k)$$

$$\text{where } x(k) = \begin{pmatrix} IV(k) \\ CS(k) \\ NI(k) \end{pmatrix}, B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, u(k) = GE(k).$$

(Note that E is singular).

For details, see Luenberger (1979, pp. 122-123).

Example 5.2.10 Consider another electric circuit given in the following diagram driven by a voltage source $v(t)$.

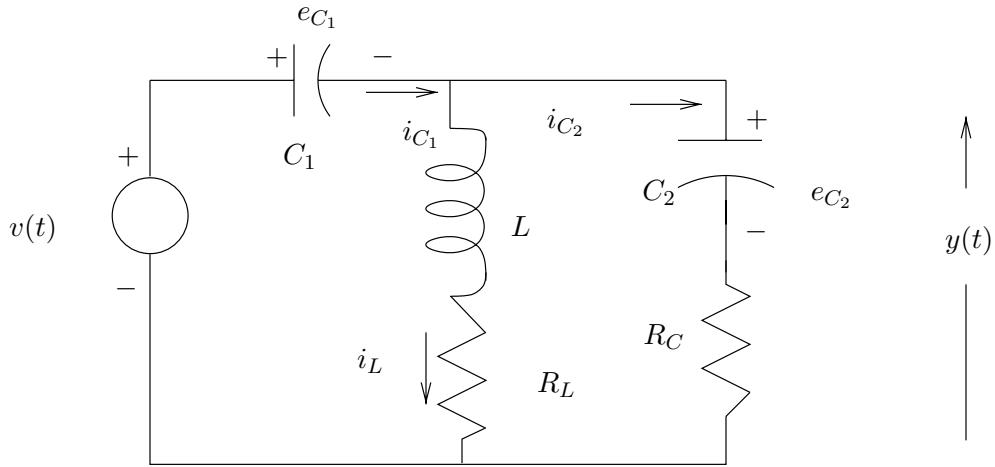


Figure 5.8 An Electric Circuit for a Descriptor System

The state variables are taken as $x_1 := e_{C_1}$, $x_2 := i_L$ and $x_3 := e_{C_2}$. By applying Kirchhoff's current and voltage laws we have

$$\begin{aligned} i_{C_1} &= i_L + i_{C_2}; \quad i_{C_1} = C_1 \frac{de_{C_1}}{dt}; \quad i_{C_2} = C_2 \frac{de_{C_2}}{dt} \\ v(t) &= e_{C_1} + L \frac{di_L}{dt} + R_L i_L = e_{C_1} + e_{C_2} + R_C i_{C_2} \end{aligned}$$

Manipulating these equations we have the state equation

$$E\dot{x} = Ax + bu$$

where $u := v$, $E = \text{diag}(R_C C_1, L, R_C C_2)$

$$x = [x_1, x_2, x_3]^T$$

$$A = \begin{bmatrix} -1 & R_C & -1 \\ -1 & -R_L & 0 \\ -1 & 0 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

The output is defined by $y(t) = e_{C_2} + R_C i_{C_2}$, so the output equation becomes

$$y = cx + du$$

where

$$c = (-1 \ 0 \ 0), \ d = 1.$$

5.3 Solutions of a Continuous-Time System: System Responses

Theorem 5.3.1 (Continuous-Time State-space Solution) *The solutions of the continuous-time dynamical equations*

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0. \quad (5.3.1)$$

$$y(t) = Cx(t) + Du(t) \quad (5.3.2)$$

are given by

$$x(t) = e^{A(t-t_0)}x_0 + \int_{t_0}^t e^{A(t-s)}Bu(s)ds \quad (5.3.3)$$

$$y(t) = Ce^{A(t-t_0)}x_0 + \int_{t_0}^t Ce^{A(t-s)}Bu(s)ds + Du(t). \quad (5.3.4)$$

Remarks: If $u(t) = 0$, then

$$x(t) = e^{A(t-t_1)}x(t_1)$$

for every $t \geq t_0$ and any $t_1 \geq t_0$.

Definition 5.3.1 *The matrix $e^{A(t-t_1)}$ is called the state transition matrix.*

Since the state at any time can be obtained from that at any other time through the state transition matrix, **it will be assumed, without any loss of generality, that $t_0 = 0$, unless otherwise mentioned.**

Assuming $t_0 = 0$, the equations (5.3.3) and (5.3.4) will, respectively, be reduced to

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-s)}Bu(s)ds \quad (5.3.5)$$

and

$$y(t) = Ce^{At}x_0 + \int_0^t Ce^{A(t-s)}Bu(s)ds + Du(t) \quad (5.3.6)$$

Definition 5.3.2 *The exponential matrix e^{At} is defined as*

$$e^{At} = \sum_{k=0}^{\infty} \frac{(At)^k}{k!}$$

Proof: Proof of (5.3.5) and (5.3.6): Noting that $\frac{d}{dt}(e^{At}) = Ae^{At}$ (see **Section 5.3.1**), we first verify that the expression (5.3.5) satisfies (5.3.1), with $t = 0$. Differentiating (5.3.5), we have

$$\begin{aligned}\dot{x}(t) &= Ae^{At}x_0 + Bu(t) + \int_0^t \frac{d}{dt}e^{A(t-s)}Bu(s)ds \\ &= Ae^{At}x_0 + Bu(t) + A \int_0^t e^{A(t-s)}Bu(s)ds \\ &= A \left[e^{At}x_0 + \int_0^t e^{A(t-s)}Bu(s)ds \right] + Bu(t) \\ &= Ax(t) + Bu(t).\end{aligned}$$

Also, note that at $t = 0$,

$$x(0) = x_0.$$

Thus the solution $x(t)$ also satisfies the initial condition.

The expression for $y(t)$ in (5.3.6) follows immediately by substituting the expression for $x(t)$ from (5.3.5) into $y(t) = Cx(t) + Dx(t)$. ■

Free, Forced and Steady-state Responses

Given the initial condition x_o and the control input $u(t)$, the vectors $x(t)$ and $y(t)$ determine the **system time responses** for $t \geq 0$. The system time responses determine the behavior of the dynamical system for particular classes of inputs. System characteristics such as **overshoot**, **rise-time**, **settling time**, etc. can be determined from the time responses.

In the expression (5.3.6), the first term $Ce^{At}x_o$ represents the response of the system due to the initial condition x_0 with zero input (**zero-input response**). This is called the **free response** of the system.

On the other hand, the second term in (5.3.6) determines, what is known, as the **forced response** of the system. It is due to the forcing function $u(t)$ applied to the system with zero initial conditions. A special case of the forced response is known as the **impulse response** which is obtained from (5.3.6) by setting $x_0 = 0$ and $u(t) = \delta(t)$, where $\delta(t)$ is the unit impulse or **Dirac delta function**. Then

$$\begin{aligned}y(t) &= \int_0^t (Ce^{A(t-s)}B + D\delta(t-s))u(s)ds \\ &= \int_0^t H(t-s)u(s)ds\end{aligned}\tag{5.3.7}$$

where the matrix $H(t)$, **the impulse response matrix**, is defined by

$$H(t) := Ce^{At}B + D\delta(t).\tag{5.3.8}$$

In fact, if $x_0 = 0$, then the (i, j) th element of the impulse response matrix $H(t)$ of the system (5.3.1)-(5.3.2) is the response at time t at the output i due to a unit impulse applied at time

$t = 0$ at the input j of the system, while all other inputs are kept at zero. Similarly, the **unit step response** is defined to be the output using the input as the unit step function in the manner done for an impulse response, assuming again that the initial state is zero; that is $x_0 = 0$.

A unit step function $1_+(t)$ is given by

$$1_+(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

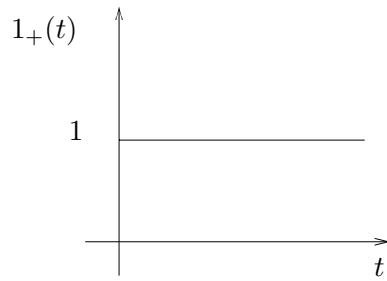


Figure 5.9 A unit step function.

For any finite value of time t , the response $y(t)$, i.e., the right-hand side of (5.3.6), will contain terms consisting of $e^{\alpha_i t} e^{j\omega_i t}$ if $\lambda_i = \alpha_i + j\omega_i$, $j = \sqrt{-1}$, is a **simple eigenvalue** of A , and the other terms are governed by the nature of the input function $u(t)$.

When t is finite, the part of the response in $y(t)$ which is governed by $e^{\alpha_i t} e^{j\omega_i t}$ is called the **transient response** of the system. As t tends to infinity, this transient part of the response tends to zero if all α_i 's are negative or it grows to become unbounded if at least one of α_i 's is positive. Thus $y_{ss}(t) \equiv \lim_{t \rightarrow \infty} y(t)$ will be called the **steady state response** of the system. The speed with which the response $y(t)$ will reach the steady state value $y_{ss}(t)$ will be determined by the largest value of α 's.

MATLAB Note. MATLAB functions **step**, **impulse** and **initial** in MATLAB CONTROL TOOLBOX can be used to generate plots for step, impulse and initial condition responses, respectively. Their syntax are:

```
step (sys)
impulse (sys)
initial (sys, x0)
```

Example 5.3.1 (Baldwin (1961), pp. 29-44).

The dynamic behavior of a moving coil galvanometer, see Fig. 5.10,

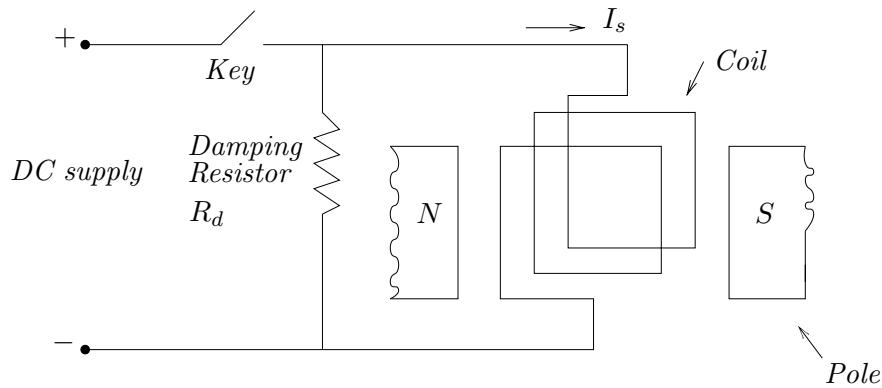


Figure 5.10 Basic Circuit of a Moving Coil Galvanometer.

is governed by

$$J \frac{d^2\theta}{dt^2} + D \frac{d\theta}{dt} + C\theta = GI_s \quad (5.3.9)$$

where

J = The moment of inertia about the axis of rotation of moving parts of the instrument

D = The damping constant

C = Stiffness of suspension

G = Galvanometer constant which represents the electromagnetic torque developed on the coil by one ampere of current flowing through it

θ = The deflection of the coil about its axis of rotation

I_s = The steady-state current flowing through the galvanometer coil

R_g = galvanometer resistance.

It can be shown that D is given by

$$D = \frac{G^2}{R_g + R_d} + D_{air},$$

where

R_g = Resistance of galvanometer coil,

R_d = Damping resistor

D_{air} = Damping to the coil due to air

If the key is opened interrupting supply current I_s to the galvanometer, the response of the coil is determined by (5.3.9) with $I_s = 0$ and is shown in Fig. 5.11 where θ_0 is the steady state deflection with I_s flowing through the coil.

A galvanometer with a very low damping constant is known as a ballistic galvanometer. If a charged capacitor is discharged through a ballistic galvanometer such that the whole charge should have passed before the coil has time to move appreciably, we have the **torque impulse** due to the whole charge equal to $\int Gidt = GQ$, the integral being taken over the time of passage of the charge Q and i is the instantaneous current flowing through galvanometer coil. The subsequent time response of the galvanometer will be similar to that shown in Fig. 5.11 but differing in the fact that the response now starts from the origin. The responses in three cases: damped, undamped, and critically damped, are shown in the figure below.

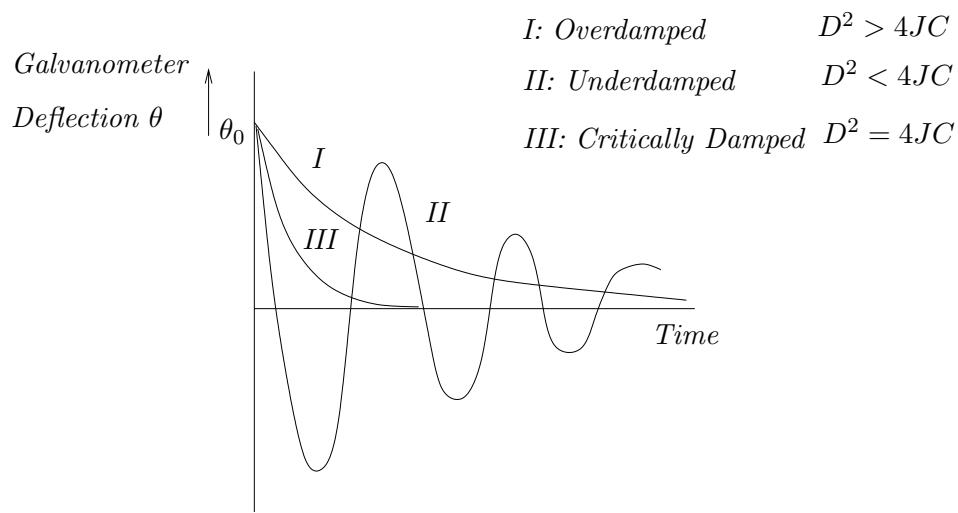


Figure 5.11 Step-Response of the Galvanometer

Causality. If the output of the system at time t does not depend upon the input applied to the system after time t , but depends only upon the present and the past inputs, the system is said to be **causal**.

In this book, all systems will be assumed to be causal.

5.3.1 Some Important Properties of the Matrix e^{At}

Since the exponential matrix e^{At} plays a fundamental role in the solution of the state equations, we will now discuss the various methods for computing this matrix. Before doing that, we list some important properties of this matrix. These properties are easily verifiable and left as [Exercises 9-11] for the readers.

1. $e^{A(t+s)} = e^{At} \cdot e^{As}$
2. $\frac{d}{dt}(e^{At}) = Ae^{At} = e^{At}A$
3. $e^{(A+B)t} = e^{At} \cdot e^{Bt}$, if and only if A and B commute; that is, if and only if $AB = BA$.
4. e^{At} is nonsingular and $(e^{At})^{-1} = e^{-At}$.

5. $\left(e^{\frac{A}{m}}\right)^m = e^A$ where m is an integer.

6. $e^{P^{-1}APt} = P^{-1}e^{At}P$

5.3.2 Sensitivity of e^{At}

We know that the accuracy of a solution obtained by an algorithm is greatly influenced by the sensitivity of the problem. We will, therefore, consider the sensitivity aspect of the problem of computing e^{At} first. We just state a result due to Van Loan (1977) without proof, which will help identify the condition number of the problem.

Let E be a perturbation matrix. Then we are interested in knowing how large the **relative error**

$$\rho = \frac{\|e^{(A+E)t} - e^{At}\|_2}{\|e^{At}\|_2}$$

can be.

Differentiating $e^{(A+E)s}e^{A(t-s)}$ with respect to s , we obtain

$$e^{(A+E)t} - e^{At} = \int_0^t e^{A(t-s)} E e^{(A+E)s} ds.$$

It then follows that

$$\rho \leq \frac{\|E\|_2}{\|e^{At}\|_2} \int_0^t \|e^{A(t-s)}\|_2 \|e^{(A+E)s}\|_2 ds$$

Further simplification of this result is possible.

Van Loan (1977) has shown that, for a given t , there exists a perturbation matrix E such that

$$\rho = \frac{\|e^{(A+E)t} - e^{At}\|_2}{\|e^{At}\|_2} \approx \kappa(A, t) \frac{\|E\|_2}{\|A\|_2},$$

where

$$\kappa(A, t) = \max_{\|E\|_2 \leq 1} \left\| \int_0^t e^{A(t-s)} E e^{As} ds \right\|_2 \frac{\|A\|_2}{\|e^{At}\|_2}$$

This result shows that $\kappa(A, t)$ is the condition number for the problem e^{At} . If this number is large, then a small change in A can cause a large change in e^{At} , for a given t .

Though determining $\kappa(A, t)$ involves computation of a matrix integral it can be easily verified that

$$\kappa(A, t) \geq t \|A\|_2,$$

with equality holding for all nonnegative t if and only if A is a normal matrix, that is, if $A^T A = A A^T$. “When A is not normal, $\kappa(A, t)$ can grow like a high degree polynomial in t ” (Moler and Van Loan (1978)).

Example 5.3.2 Consider computing e^A , where

$$A = \begin{pmatrix} -1 & 1000 \\ 0 & -1 \end{pmatrix}.$$

Since $\| A \|_2 = 10^3$, the problem of computing the matrix exponential e^A is expected to be ill-conditioned.

Let's verify this as follows. The matrix e^A computed by using the MATLAB function **expm** is

$$e^A = \begin{pmatrix} 0.3679 & 367.8794 \\ 0 & 0.3679 \end{pmatrix}.$$

Now change the (2, 1) entry of A to 10^{-4} and call this perturbed matrix A_{new} . We now obtain

$$e^{A_{new}} = \begin{pmatrix} 0.3496 & 361.7787 \\ 0 & 0.3496 \end{pmatrix}.$$

Note that each nonzero entry of e^A has now changed.

The relative error in the solution: $\frac{\| e^A - e^{A_{new}} \|_2}{\| e^A \|_2} = 0.0166$. On the other hand, the relative error in the data: $\frac{\| A - A_{new} \|_2}{\| A \|_2} = 10^{-7}$.

(The matrix $e^{A_{new}}$ was also computed by the MATLAB function **expm**).

5.3.3 Computational Methods for e^{At}

There are a wide variety of methods to compute the exponential matrix. Several of these methods have been carefully analyzed, with respect to efficiency, and numerical stability, in an authoritative paper on the subject by Moler and Van Loan (1978). The methods can be broadly classified as:

- I. The Eigenvalue-Eigenvector Method
- II. Series Methods
- III. ODE Methods (Ordinary Differential Equations Methods)
- IV. Matrix Decomposition Methods

We will discuss below some of these methods.

The Eigenvalue-Eigenvector Method

We have seen that the solution of the unforced system:

$$\dot{x}(t) = Ax(t), x(0) = x_0 \quad (5.3.10)$$

is given by

$$x(t) = e^{At}x_0. \quad (5.3.11)$$

Equation (5.3.11) shows that the i -th column of the matrix e^{At} is just the vector $x(t)$ with $x(0) = e_i$, the i -th unit vector.

Again, $x(t)$ can be expressed in terms of the eigenvalues and eigenvectors of A :

$$x(t) = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2 + \cdots + c_n e^{\lambda_n t} v_n, \quad (5.3.12)$$

where $\lambda_1, \lambda_2, \dots, \lambda_n$ are the **simple eigenvalues of A** and v_1 through v_n are a set of linearly independent eigenvectors associated with λ_1 through λ_n . The scalar c 's are computed from the given initial condition.

Thus, when the eigenvalues of A are simple, the matrix e^{At} is completely determined by the eigenvalues and eigenvectors of the matrix A . The same can be shown to be true when A has multiple eigenvalues.

A difficulty with this approach arises when A has some nearly equal eigenvalues. This can be seen from the following theorem (Moler and Van Loan (1978)).

Theorem 5.3.2 *Let A be an $n \times n$ nondefective matrix; that is, it has a set of n linearly independent eigenvectors. Let $X^{-1}AX = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$, where $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of A . Then*

$$\|fl(e^{At}) - e^{At}\|_2 \leq n\mu e^{\rho(A)t} \text{Cond}_2(X),$$

where $\rho(A) = \max |\lambda_i|$ is the spectral radius of A .

Interpretation of Theorem 5.3.2: Theorem 5.3.2 shows that **there might be a large error in computing e^{At} whenever there is a coalescence of eigenvalues of A** , because, in this case, $\text{Cond}_2(X)$ will be large.

The following simple 2×2 example taken from Moler and Van Loan (1978) illustrates the difficulty.

Let

$$A = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}.$$

Then

$$e^{At} = \begin{pmatrix} e^{\lambda t} & \alpha \frac{e^{\lambda t} - e^{\mu t}}{\lambda - \mu} \\ 0 & e^{\mu t} \end{pmatrix}.$$

Clearly, the result will be inaccurate if λ is close to μ , but not exactly equal to μ . A large round-off error can be expected in this case.

Series Method for Computing the Exponential Matrix

In this section, we briefly state two series methods: **The Taylor Series Method** and the **Padé Approximation Method**. When properly implemented, these methods become numerically effective for computing the matrix exponential matrix.

The Taylor Series Method

An obvious way to approximate e^A is to evaluate a finite-series sum by truncating the Taylor series

$$e^A = I + A + \frac{A^2}{2} + \frac{A^3}{6} + \dots$$

after k terms. Thus, if

$$T_k(A) = \sum_{j=0}^k \frac{A^j}{j!}$$

and if $fl(T_k(A))$ denotes the floating point evaluation of $T_k(A)$, then it is natural to choose k so that $fl(T_k(A)) = fl(T_{k+1}(A))$. **The drawbacks to this method are that a large number of terms is needed for convergence, and even when convergence occurs, the answer can be totally wrong.**

Consider the following example from Moler and Van Loan (1978).

Example 5.3.3 Consider computing e^A using the Taylor series methods with

$$A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix}$$

A total of $k = 59$ terms were required for convergence and the computed output was

$$e^A \approx \begin{pmatrix} -22.25880 & -1.432766 \\ -61.49931 & -3.474280 \end{pmatrix},$$

which is nowhere close to the true answer (to 6 decimal places)

$$e^A \approx \begin{pmatrix} -0.735759 & 0.551819 \\ -1.471518 & 1.103638 \end{pmatrix}.$$

The source of error here is the catastrophic cancellation that took place in the evaluation of $\frac{A^{16}}{16!} + \frac{A^{17}}{17!}$, using finite-precision arithmetic (see **Chapter 3**). These two terms have almost the same magnitude but are of opposite signs, as seen from the following expressions of $\frac{A^{16}}{16!}$ and $\frac{A^{17}}{17!}$:

$$\frac{A^{16}}{16!} = 10^6 \begin{pmatrix} 6.9773 & -3.4886 \\ 9.3030 & -4.6515 \end{pmatrix}.$$

$$\frac{A^{17}}{17!} = 10^6 \begin{pmatrix} -6.9772 & 3.4885 \\ -9.3030 & 4.6515 \end{pmatrix}.$$

The Padé Approximation Method

Suppose that $f(x)$ is a power series represented by

$$f(x) = f_0 + f_1x + f_2x^2 + \dots$$

then the (p, q) Padé approximation to $f(x)$ is defined as

$$f(x) \approx \frac{c(x)}{d(x)} = \frac{\sum_{k=0}^p c_k x^k}{\sum_{k=0}^q d_k x^k}$$

The coefficients $c_k, k = 0, \dots, p$, and $d_k, k = 0, \dots, q$ are chosen such that the terms containing $x^0, x^1, x^2, \dots, x^{p+q}$ are cancelled out in $f(x)d(x) - c(x)$. The order of the Padé approximation is $p+q$. The ratio $\frac{c(x)}{d(x)}$ is unique and the coefficients c_1, c_1, \dots, c_p and d_0, d_1, \dots, d_q always exist.

The (p, q) Padé approximation to e^A is given by

$$R_{pq}(A) = [D_{pq}(A)]^{-1} N_{pq}(A),$$

where

$$D_{pq}(A) = \sum_{j=0}^q \frac{(p+q-j)!q!}{(p+q)!j!(q-j)!} (-A)^j$$

and

$$N_{pq}(A) = \sum_{j=0}^p \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!} A^j.$$

It can be shown [Exercise 18] that if p and q are large, or if A is a matrix having all its eigenvalues **with negative real parts**, then $D_{pq}(A)$ is nonsingular.

Round-off errors due to catastrophic cancellation is again a major concern for this method. It is less when $\|A\|$ is not too large and the diagonal approximants ($p = q$) are used.

Scaling and Squaring in Taylor Series and Padé Approximations

The difficulties with the round-off errors in Taylor series and Padé approximation methods can somehow be controlled by a technique known as **Scaling and Squaring**. Since $e^A = (e^{\frac{A}{m}})^m$, the idea here is to choose m to be a power of 2 so that $e^{\frac{A}{m}}$ can be computed rather efficiently and accurately using the method under consideration, and then to form the matrix $(e^{\frac{A}{m}})^m$ by repeated squaring.

Moler and Van Loan (1978) have remarked “**When properly implemented, the resulting algorithm (Taylor series or Padé approximation method with scaling and squaring) is one of the most effective we know.**” The method may fail but it is **reliable** in the sense that it tells the user when it does.

The method has favorable numerical properties when $p = q$, we will, therefore, describe the algorithm for this case only.

Let $m = 2^j$ be chosen such that $\|A\|_\infty \leq 2^{j-1}$, then Moler and Van Loan (1978) have shown that there exists an E such that

$$[R_{pp}(A/2^j)]^{2^j} = e^{A+E}$$

where $\|E\|_\infty \leq \epsilon \|A\|_\infty$, with $\epsilon = 2^{3-2p} \left(\frac{(p!)^2}{(2p)!(2p+1)!} \right)$.

Given an error-tolerance δ , the above expression, therefore, gives a criterion to choose p such that $\|E\|_\infty \leq \delta \|A\|_\infty$.

The above discussion leads to the following algorithm.

Algorithm 5.3.1 Padé Approximation to e^A using Scaling and Squaring

Input: $A \in \mathbb{R}^{n \times n}$, $\delta > 0$, an error-tolerance.

Output: $F = e^{A+E}$ with $\|E\|_\infty \leq \delta \|A\|_\infty$.

Step 1. Choose j such that $\|A\|_\infty \leq 2^{j-1}$. Set $A \equiv A/2^j$.

Step 2. Find p such that p is the smallest non-negative integer satisfying

$$\left(\frac{8}{2^{2p}}\right) \frac{(p!)^2}{(2p)!(2p+1)!} \leq \delta.$$

Step 3. Set $D \equiv I$, $N \equiv I$, $Y \equiv I$, $c = 1$.

Step 4. For $k = 1, 2, \dots, p$ do

$$c \equiv c(p-k+1)/[(2p-k+1)k]$$

$$Y \equiv AY, N \equiv N + cY, D = D + (-1)^k cY.$$

End

Step 5. Solve for F : $DF = N$.

Step 6. For $k = 1, 2, \dots, j$ do

$$F \equiv F^2.$$

End

Flop-count: · The algorithm requires about $2(p+j+\frac{1}{3})n^3$ flops.

Numerical Stability Property:

The algorithm is numerically stable when A is normal. When A is non-normal, an analysis of the stability property becomes difficult, because, in this case e^A may grow before it decays during the squaring process; which is known as “hump” phenomenon. For details, see Golub and Van Loan (1996, 576).

MATLAB Note: The MATLAB function `expm` computes the exponential of a matrix A , using Algorithm 5.3.1.

MATCONTROL Note: Algorithm 5.3.1 has been implemented in MATCONTROL function `expmpade`.

Example 5.3.4 Consider computing e^A using Algorithm 5.3.1 with

$$A = \begin{pmatrix} 5 & 1 & 0 \\ 0 & 2 & 0 \\ 2 & 3 & 1 \end{pmatrix}.$$

Set $\delta = 0.50$.

Step 1. Since $\|A\|_\infty = 7$, choose $j = 4$.

Then

$$A \equiv \frac{A}{2^4} = \begin{pmatrix} 0.3125 & 0.8625 & 0 \\ 0 & 0.1750 & 0 \\ 0.1250 & 0.1875 & 0.6625 \end{pmatrix}$$

Step 2. $p = 1$.

Step 3. $D = N = Y = I$.

Step 4. $k = 1, c = 0.5$.

$$Y = \begin{pmatrix} 0.3125 & 0.0625 & 0 \\ 0 & 0.1250 & 0 \\ 0.1250 & 0.7875 & 0.0625 \end{pmatrix}$$

$$N = \begin{pmatrix} 1.1563 & 0.0313 & 0 \\ 0 & 1.0625 & 0 \\ 0.0625 & 0.09838 & 1.0313 \end{pmatrix}$$

$$D = \begin{pmatrix} 0.8438 & -0.0313 & 0 \\ 0 & 0.9375 & 0 \\ -0.0625 & -0.0938 & 0.9638 \end{pmatrix}$$

$$F = \begin{pmatrix} 1.3704 & 0.0790 & 0 \\ 0 & 1.1333 & 0 \\ 0 & 0.2115 & 1.0645 \end{pmatrix}$$

$$\text{Step 5. } F \equiv F^2 = \begin{pmatrix} 154.6705 & 49.0874 & 0 \\ 0 & 7.4084 & 0 \\ 75.9756 & 36.2667 & 2.7192 \end{pmatrix}$$

The matrix e^A given by F in Step 5 is different from what would have been obtained by using MATLAB function $\text{expm}(A)$:

$$e^A = \begin{pmatrix} 148.4132 & 47.0080 & 0 \\ 0 & 7.3891 & 0 \\ 72.8474 & 35.1810 & 2.7183 \end{pmatrix}$$

■

This is because MATLAB uses much smaller tolerance than what was prescribed for this example.

ODE Methods.

Consider solving

$$\dot{x}(t) = f(x, t), x(0) = x_0$$

with

$$f(x, t) = Ax(t).$$

Then the k-th column of e^{At} becomes equal to the solution $x(t)$ by setting $x(0)$ as the k-th column of the identity matrix. Thus, any ODE solver can be used to compute e^{At} .

However, computing e^{At} using a general-purpose ODE solver will be rather expensive, because such a method can not take advantage of the special form of $f(x, t) = Ax(t)$. **An ODE routine will treat $Ax(t)$ as any function $f(x, t)$ and the computation will be carried on.**

However, a single-step ODE method such as the fourth order **Taylor** or **Runge-Kutta** method and a multistep solver such as the **Adams formulas** with variable step size, could be rather computationally practically feasible (and also reliable and stable), for the matrix vector problem of computing $e^{At}x(0)$, when such problem is to be solved for many different values of t and also when A is large and sparse. ■

Matrix Decomposition Methods

The basic idea behind such methods is to first transform the matrix A to a suitable canonical form R so that e^R can be easily computed, and then compute e^A from e^R . Thus, if P is the transforming matrix such that

$$P^{-1}AP = R,$$

then $e^A = Pe^RP^{-1}$.

The convenient choices for R include the **Jordan canonical form**, **the companion form**, **the Hessenberg form**, and **the real Schur form of a matrix**. Because of the difficulties in using the Jordan canonical form and the companion form, stated in section 4.1, we will not discuss computing e^A via these forms here.

Though the Hessenberg form can be obtained in a numerically stable way, computation of e^H via an upper Hessenberg matrix H will involve divisions by the superdiagonal entries, and if the product of these entries is small, large round-off errors can contaminate the computation (see **Exercise 13**). Thus, **our choice for R here is the Real Schur form**.

Computing e^A via the Real Schur Form.

Let A be transformed to a real Schur matrix R using an orthogonal similarity transformation:

$$P^TAP = R,$$

then $e^A = Pe^R P^T$.

The problem is now how to compute e^R . Parlett (1976) has given an elegant formula to compute $f(R)$, for an analytic function $f(x)$, where R is **upper triangular**. The formula is derived from the commutativity property: $Rf(R) = f(R)R$.

Adapting this formula for computing $F = e^R$, we have the following algorithm when R is an upper triangular matrix. The algorithm needs to be modified when R is a **quasi-triangular** matrix [Exercise 23].

Algorithm 5.3.2 The Schur Algorithm for e^A .

Input: $A \in \mathbb{R}^{n \times n}$

Output: e^A .

Step 1. Transform A to R an **upper triangular** matrix using the QR iteration algorithm (Chapter 4):

$$P^T AP = R.$$

(Note that when the eigenvalues of A are all real, the real Schur form is upper triangular).

Step 2. Compute $e^R = G = (g_{ij})$:

For $i = 1, \dots, n$ do

$$g_{ii} = e^{r_{ii}}$$

End

For $k = 1, 2, \dots, n-1$ do

For $i = 1, 2, \dots, n-k$ do

Set $j = i + k$

$$g_{ij} = \frac{1}{(r_{ii} - r_{jj})} \left[r_{ij}(g_{ii} - g_{jj}) + \sum_{p=i+1}^{j-1} (g_{ip}r_{pj} - r_{ip}g_{pj}) \right].$$

End

End

Step 3. Compute $e^A = Pe^R P^T$

Flop-count. Computation of e^R in Step 2 requires about $\frac{2n^3}{3}$ flops.

MATCONTROL Note: Algorithm 5.3.2 has been implemented in MATCONTROL function `expmschr`.

Example 5.3.5 Consider computing e^A using Algorithm 5.3.2 with the matrix A of Example 5.3.3.

$$A = \begin{pmatrix} 5 & 1 & 0 \\ 0 & 2 & 0 \\ 2 & 3 & 1 \end{pmatrix}.$$

Using MATLAB function $[P, R] = \text{schur}(A)$, we obtain $P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$, and

$$R = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 1 \\ 0 & 0 & 2 \end{pmatrix}.$$

$$\begin{aligned} g_{11} &= e^{r_{11}} = 2.7183, \quad g_{22} = e^{r_{22}} = 148.4132, \quad g_{33} = e^{r_{33}} = 7.3891 \\ \mathbf{k} = \mathbf{1}, i = 1, j = 2 : \quad g_{12} &= \frac{1}{(r_{11} - r_{22})}[r_{12}(g_{11} - g_{22})] = 72.8474 \\ \mathbf{k} = \mathbf{1}, i = 2, j = 3 : \quad g_{23} &= \frac{1}{(r_{22} - r_{33})}[r_{23}(g_{22} - g_{33})] = 47.0090 \\ \mathbf{k} = \mathbf{2}, i = 1, j = 3 : \quad g_{13} &= \frac{1}{(r_{11} - r_{33})}[r_{13}(g_{11} - g_{33}) + (g_{12}r_{23} - r_{12}g_{23})] = 35.1810 \end{aligned}$$

So,

$$e^R = \begin{pmatrix} 2.7183 & 72.8474 & 35.1810 \\ 0 & 148.4132 & 47.0080 \\ 0 & 0 & 7.3891 \end{pmatrix}.$$

Thus

$$e^A = Pe^R P^T = \begin{pmatrix} 148.4132 & 47.0080 & 0 \\ 0 & 7.3891 & 0 \\ 72.8474 & 35.1810 & 2.7183 \end{pmatrix}.$$

Note that e^A obtained here is exactly the same (in four digit arithmetic) as given by MATLAB function **expm** (A), which is based on Algorithm 5.3.1.

Numerical Stability of the Schur Algorithm: Numerical difficulties clearly arise when A has equal or nearly equal confluent eigenvalues, even though the transformation matrix P is orthogonal.

5.3.4 Comparison of Different Methods for Computing the Exponential Matrix

From our discussions in this section, it is clear that the **Padé approximation method** (with scaling and squaring) and the **Schur method** should, in general, be attractive from computational viewpoints.

The **Taylor series method** and the methods based on reduction of A to a **companion matrix** or to the **Jordan Canonical form** should be avoided for the reason of **numerical instability**. The ODE techniques should be preferred when the matrix A is large and sparse.

5.3.5 Evaluating an Integral with the Exponential Matrix

We have discussed methods for computing the exponential matrix. We now present a method due to Van Loan (1978) to compute integrals involving exponential matrices.

The method can be used, in particular, to compute the state-space solution (5.3.3) of the equation (5.3.1), and the **controllability** and **observability Grammians**, which will be discussed in the next chapter.

The method uses diagonal Padé approximation discussed in the last section.

Let

$$\begin{aligned} H(\Delta) &= \int_0^\Delta e^{As} B ds, \\ N(\Delta) &= \int_0^\Delta e^{A^T s} Q e^{As} ds, \\ M(\Delta) &= \int_0^\Delta e^{A^T s} Q H(s) ds, \\ W(\Delta) &= \int_0^\Delta H(s)^T Q H(s) ds \end{aligned}$$

where A and B are matrices of order n and $n \times m$ respectively, and Q is a symmetric positive semidefinite matrix.

Algorithm 5.3.3 An Algorithm for computing Integrals involving Matrix Exponential

Inputs:

- A – The $n \times n$ state matrix
- B – An $n \times m$ matrix
- Q – A symmetric positive semidefinite matrix.

Outputs: F, H, Q, M , and W ; which are, respectively, the approximations to $e^{A\Delta}$, $H(\Delta)$, $N(\Delta)$, $M(\Delta)$, and $W(\Delta)$.

Step 1. Form the $(3n + m) \times (3n + m)$ matrix

$$\hat{C} = \begin{pmatrix} -A^T & I & 0 & 0 \\ 0 & -A^T & Q & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Find the smallest positive integer j such that $(\|\hat{C}\Delta\|_F/2^j) \leq \frac{1}{2}$. Set $t_0 = \frac{\Delta}{2^j}$.

Step 2. For some $q \geq 1$, compute

$$Y_0 = R_{qq}(\frac{\hat{C}\Delta}{2^j}),$$

where R_{qq} is the (q, q) Padé approximant to e^z :

$$R_{qq}(z) = \frac{\sum_{k=0}^z c_k z^k}{\sum_{k=0}^q c_k (-z)^k}, \text{ where } c_k = \frac{(2q-k)!q!}{(2q)!k!(q-k)!}$$

Write

$$Y_0 = \begin{pmatrix} F_1(t_0) & G_1(t_0) & H_1(t_0) & K_1(t_0) \\ 0 & F_2(t_0) & G_2(t_0) & H_2(t_0) \\ 0 & 0 & F_3(t_0) & G_3(t_0) \\ 0 & 0 & 0 & F_4(t_0) \end{pmatrix}$$

and set

$$\begin{aligned} F_0 &= F_3(t_0) \\ H_0 &= G_3(t_0) \\ Q_0 &= F_3(t_0)^T G_2(t_0) \\ M_0 &= F_3(t_0)^T H_2(t_0) \\ W_0 &= [B^T F_3(t_0)^T K_1(t_0)] + (B^T F_3(t_0)^T K_1(t_0))^T. \end{aligned}$$

Step 3. For $k = 0, 1, \dots, j-1$ do

$$W_{k+1} = 2W_k + H_k^T M_k + M_k^T H_k + H_k^T Q_k H_k$$

$$M_{k+1} = M_k + F_k^T [Q_k H_k + M_k]$$

$$Q_{k+1} = Q_k + F_k^T Q_k F_k$$

$$H_{k+1} = H_k + F_k H_k$$

$$F_{k+1} = F_k^2$$

End

Step 4. Set $F \equiv F_j$, $H \equiv H_j$, $Q \equiv Q_j$, $M \equiv M_j$, and $W \equiv W_j$.

Remark: It has been proved by Van Loan (1978) that the accuracy of the algorithm can be controlled by selecting q properly. For “**how to choose q properly**” and other computational details, see the paper of Van Loan (1978).

MATCONTROL Note: Algorithm 5.3.3 has been implemented in MATCONTROL function `intmexp`.

5.4 State-Space Solution of the Discrete-Time System

In this section, we state a discrete-analog of Theorem 5.3.1, and then discuss how to compute the discrete-time solution.

Theorem 5.4.1 (Discrete-Time State-space Solution)

The solutions to the linear time-invariant discrete-time system

$$x(k+1) = Ax(k) + Bu(k), x(0) = x_0 \quad (5.4.1)$$

and

$$y(k) = Cx(k) + Du(k) \quad (5.4.2)$$

are

$$x(k) = A^k x_0 + \sum_{i=0}^{k-1} A^{k-1-i} Bu(i) \quad (5.4.3)$$

and

$$y(k) = CA^k x_0 + \left\{ \sum_{i=0}^{k-1} CA^{k-i-1} Bu(i) \right\} + Du(k) \quad (5.4.4)$$

Proof: From

$$x(k+1) = Ax(k) + Bu(k), \quad (5.4.5)$$

we have

$$\begin{aligned} x(k) &= A[Ax(k-2) + Bu(k-2)] + Bu(k-1) \\ &= A^2x(k-2) + ABu(k-2) + Bu(k-1) \\ &= A^2[Ax(k-3) + Bu(k-3)] + ABu(k-2) + Bu(k-1) \\ &\vdots \\ &= A^k x_0 + \sum_{i=0}^{k-1} A^{k-1-i} Bu(i) \end{aligned}$$

This proves (5.4.3).

The equation (5.4.4) is now obtained by substituting the expression of $x(t)$ from (5.4.3) into (5.4.2). ■

Computing the Powers of a Matrix

Theorem 5.4.1 shows that, to find the state-space solution of a discrete-time system, one needs to compute the various powers of A .

The powers of a matrix A are more easily computed if A is first decomposed into a condensed form by similarity.

Thus if $P^T AP = R$, where P is orthogonal, then $A^s = PR^sP^T$.

For the sake of numerical stability, only those condensed forms such as the Hessenberg form or the real Schur form of a matrix should be considered.

The matrix R^s can be easily calculated by exploiting the Hessenberg or Schur structure of R . Furthermore, the reduction to R can be achieved using a numerically stable procedure such as Householder's or Givens' method (**Chapter 4**).

5.5 Transfer Function and Frequency Response

In this section, we introduce the important concepts of **transfer function** and **frequency response** matrices and describe a numerical algorithm for computing the frequency response matrix.

5.5.1 Transfer Function

Consider

$$\dot{x}(t) = Ax(t) + Bu(t), x(0) = x_0 \quad (5.5.1)$$

$$y(t) = Cx(t) + Du(t). \quad (5.5.2)$$

Let $\hat{x}(s)$, $\hat{y}(s)$, and $\hat{u}(s)$, respectively, denote the Laplace transforms of $x(t)$, $y(t)$, and $u(t)$. Then taking the Laplace transform of (5.5.1) and (5.5.2), we obtain

$$s\hat{x}(s) - x_0 = A\hat{x}(s) + B\hat{u}(s) \quad (5.5.3)$$

$$\hat{y}(s) = C\hat{x}(s) + D\hat{u}(s) \quad (5.5.4)$$

From (5.5.3) and (5.5.4), we have

$$\hat{x}(s) = R(s)x(0) + R(s)B\hat{u}(s) \quad (5.5.5)$$

$$\hat{y}(s) = CR(s)x(0) + G(s)\hat{u}(s), \quad (5.5.6)$$

where

$$R(s) = (sI - A)^{-1} \quad (5.5.7)$$

and

$$G(s) = C(sI - A)^{-1}B + D. \quad (5.5.8)$$

If $x(0) = 0$, then (5.5.6) gives

$$\hat{y}(s) = G(s)\hat{u}(s).$$

Definition 5.5.1 *The matrix $R(s)$ is called the **resolvent** and $G(s)$ is called the **transfer function**.*

The transfer function $G(s)$ is a matrix transfer function of dimension $r \times m$. Its (i, j) th entry denotes the transfer function from j -th input to the i -th output. That is why, it is also referred to as the **transfer function matrix** or simply the **transfer matrix**.

Definition 5.5.2 *The points p at which $G(p) = \infty$ are called the **poles** of the system.*

*If $G(\infty) = 0$, then the transfer function is called **strictly proper** and is **proper** if $G(\infty)$ is a constant matrix.*

State-Space Realization

For computational convenience, the transfer function matrix $G(s)$ will be written sometimes as $G(s) = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right]$. The state-space model (A, B, C, D) having $G(s)$ as its transfer function matrix is called a **realization** of $G(s)$. For more on this topic, see **Chapter 9**.

Operations with Transfer Function Matrices

Let $G_1(s)$ and $G_2(s)$ be the transfer functions of the two systems S_1 and S_2 . Then the transfer function matrix of the **parallel connection** of S_1 and S_2 is $G_1(s) + G_2(s)$. Using our notation above, we obtain

$$G_1(s) + G_2(s) = \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array} \right] + \left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array} \right] = \left[\begin{array}{cc|c} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ \hline C_1 & C_2 & D_1 + D_2 \end{array} \right].$$

Similarly, the transfer function matrix of the **series or cascade connection** of S_1 and S_2 (i.e., a system with the output of the second system as the input of the first system) is $G_1(s)G_2(s)$, given by

$$G_1(s)G_2(s) = \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array} \right] \left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array} \right] = \left[\begin{array}{cc|c} A_2 & 0 & B_1 \\ B_1C_2 & A_1 & B_1D_2 \\ \hline D_1C_2 & C_1 & D_1D_2 \end{array} \right].$$

The **transpose** of a transfer function matrix $G(s)$ is defined as

$$G^T(s) = B^T(sI - A^T)^{-1}C^T + D^T,$$

or equivalently,

$$G^T(s) = \left[\begin{array}{c|c} A^T & B^T \\ \hline C^T & D^T \end{array} \right].$$

The **conjugate of $G(s)$** is defined as

$$G^\sim(s) \equiv G^T(-s) = B^T(-sI - A^T)^{-1}C^T + D^T,$$

or equivalently,

$$G^\sim(s) = \left[\begin{array}{c|c} -A^T & -C^T \\ \hline -B^T & D^T \end{array} \right].$$

The **inverse** of a transfer function matrix $G(s)$, denoted by $\hat{G}(s)$ is such that $G(s)\hat{G}(s) = \hat{G}(s)G(s) = I$. If $G(s)$ is square and D is invertible, then

$$\hat{G}(s) \equiv G^{-1}(s) = \left[\begin{array}{c|c} A - BD^{-1}C & -BD^{-1} \\ \hline D^{-1}C & D^{-1} \end{array} \right].$$

MATLAB Notes. MATLAB functions **parallel**, **series**, **transpose**, **inv** (**ss/inv.m**) can be used to compute parallel, series, transpose, and inverse, respectively.

Transfer Function of Discrete-time System.

The transfer function matrix of the discrete-time system (5.4.1)-(5.4.2) is

$$G(z) = C(zI - A)^{-1}B + D.$$

It is obtained by taking the z -transform of the system.

5.5.2 The Frequency Response Matrix and its Computation

In this section, we describe a computationally viable approach for computing the **frequency response matrix**.

Definition 5.5.3 *For continuous-time state-space modal (5.5.1-5.5.2), the matrix*

$$G(j\omega) = C(j\omega I - A)^{-1}B + D \quad (5.5.9)$$

is called the frequency response matrix; $\omega \in \mathbb{R}$ is called frequency.

The frequency response matrix of a discrete-time system is similarly defined by using the transformation

$$z = e^{j\omega T},$$

where T is the sample time. This transformation maps the frequencies ω to points on the unit circle. The frequency response matrix is then evaluated at the resulting z values.

Computing the Frequency Response Matrix

In order to study the different responses of the system, it is necessary to compute $G(j\omega)$ for many different values of ω . Furthermore, the singular values of the **return difference matrix** $I + L(j\omega)$ and of the inverse return difference matrix $I + L^{-1}(j\omega)$, where $L(j\omega)$ is square and of the form $L = KGM$ for appropriate K and M , **provide robustness measure of a linear system with respect to stability, noise, disturbance attenuation, sensitivity, etc.** (Safonov et al. (1981)).

We therefore describe a numerical approach to compute $G(j\omega)$. **For simplicity, since D does not have any computational role in computing $G(j\omega)$, we will assume that $D = 0$.**

The computation of $(j\omega I - A)^{-1}B$ is equivalent to solving m systems:

$$(j\omega I - A)X = B,$$

A usual scheme for computing the frequency response matrix is:

Step 1. Solve the m systems for m columns x_1, x_2, \dots, x_m of X :

$$(j\omega I - A)x_i = b_i, \quad i = 1, 2, \dots, m$$

where b_i is the i -th column of B .

Step 2. Compute CX .

If A is $n \times n$, B is $n \times m$ ($m \leq n$), and C is $r \times n$ ($r \leq n$), and if LU factorization is used to solve the systems $(j\omega I - A)x_i = b_i, i = 1, 2, \dots, m$, then the total flop-count (**complex**) **for each ω** is about $2n^3 + 2mn^2 + 2mnr$. Note that, since the matrix $j\omega I - A$ is the same for each linear system for a given ω , the matrix $j\omega I - A$ has to be factored into LU only once, and the same factorization can be used to solve all the m systems. Since the matrix $G(j\omega)$ needs to be computed for many different values of ω , the computation in this way will be fairly expensive. Laub (1981) noted that the computational cost can be reduced significantly when $n > m$ (which is normally the case in practice), if the matrix A is initially transformed to an upper Hessenberg matrix H by orthogonal similarity, before the frequency-response computation begins. The basic observation here is that if $P^T AP = H$, an upper Hessenberg matrix, then

$$\begin{aligned} G(j\omega) &= C(j\omega I - A)^{-1}B \\ &= C(j\omega I - PHP^T)^{-1}B \\ &= C(P(j\omega I - H)P^T)^{-1}B \\ &= CP(j\omega I - H)^{-1}P^T B. \end{aligned}$$

Thus, if A is transformed initially to an upper Hessenberg matrix H , and $CP = C'$ and $P^T B = B'$ are computed once for all, then for each ω , we have the following algorithm:

Algorithm 5.5.1 A Hessenberg Algorithm for the Frequency Response Matrix

Input. A —The $n \times n$ state matrix
 ω —Frequency, a *real number*
 B —The $n \times m$ input matrix
 C —The $r \times n$ output matrix.

Output. The Frequency Response Matrix $G(j\omega) = C(j\omega I - A)^{-1}B$.

Step 1. Transform A to an upper Hessenberg matrix H :

$$P^T AP = H.$$

Step 2. Compute $B' = P^T B$

and $C' = CP$, using the factored form of P .

Step 3. Solve the m **Hessenberg** systems:

$$(j\omega I - H)x_i = b'_i, \quad i = 1, \dots, m,$$

where b'_i is the i -th column of B' .

Step 4. Compute $C'X$.

■

Flop-Count: Since the system matrices A , B , and C are real, Step 1 and Step 2 can be done using real arithmetic and require approximately $\frac{10}{3}n^3 + (n-2)(4mn+4rn)$ (real) flops. Step 3 and Step 4 require complex arithmetic and require approximately $n^2+m(n^2)+2rnm$ **complex** flops.

Comparison: For N values of ω , the Hessenberg-method require $\frac{10}{3}n^3 + 4(n-2)(m+r)n$ real $+[2mn^2+2rnm]N$ complex flops compared to $[2n^3+2mn^2+2mnr]N$ complex flops, required by the non-Hessenberg scheme.

Numerical Stability: It is possible to show (Laub and Linnemann (1986)) that if the data is well-conditioned, then the frequency response of the computed Hessenberg form is $(C + \Delta C)(j\omega I - A - \Delta A)^{-1}(B + \Delta B)$, where ΔA , ΔB , and ΔC are small. Thus, the **Hessenberg-method is stable**.

MATCONTROL Note: Algorithm 5.5.1 has been implemented in MATCONTROL function **freqresh**.

Example 5.5.1 Compute the frequency response matrix with

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 0 & 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \omega = 1.$$

Since A is already in upper Hessenberg form, Step 1 and Step 2 are skipped.

Step 3. Solve for x_1 :

$$(j\omega I - H)x_1 = b'_1 = b_1$$

$$x_1 = \begin{pmatrix} -5.000 - 0.0000i \\ 0.4000 - 0.8000i \\ +0.1000 - 0.7000i \end{pmatrix}$$

Solve for x_2 :

$$(j\omega I - H)x_2 = b'_2 = b_2$$

$$x_2 = \begin{pmatrix} -0.5000 - 0.0000i \\ 0.4000 + 0.8000i \\ 0.1000 - 0.7000i \end{pmatrix}.$$

Step 4. Compute the frequency response matrix

$$\begin{aligned} G(j\omega) &= C'X = CX \\ &= \begin{pmatrix} -0.8000 + 0.1000i & -0.8000 + 0.1000i \\ -0.8000 + 0.1000i & -0.8000 + 0.1000i \end{pmatrix}. \end{aligned}$$

Remarks:

1. A method based on a determinant identity for the computation of the frequency response matrix has been proposed by Misra and Patel (1988). The method seems to be considerably faster and at least as accurate as the Hessenberg method just described. The method uses the controller-Hessenberg and observer-Hessenberg forms which will be described in the next chapter. The Misra-Patel method for computing the frequency response matrix is based on the following interesting observation:

$$g_{lk}(j\omega) = \frac{\det(j\omega I - A + b_k c_l^T)}{\det(j\omega I - A)} - 1, \quad (5.5.10)$$

where b_k and c_l denote, respectively, the k -th and l -th columns of the matrices B and C .

2. An alternative method also based on the reduction to controller-Hessenberg form, has been given by Laub and Linnemann (see Laub and Linnemann (1986)).
3. Another recent method for the problem has been proposed by Kenney, Laub and Stubberud (1993). The method uses matrix interpolation techniques and seems to have better efficiency than the Hessenberg method.
4. The frequency response of a discrete time system is obtained by evaluating the transfer function $H(z)$ in (5.5.8) on the unit circle.

Bode Diagram and the Singular Value Plot

Traditionally, Bode diagram is used to measure the magnitude and angle of frequency response of a SISO system. Thus expressing the frequency response function in polar coordinates, we have

$$G(j\omega) = M(\omega)e^{j\sigma(\omega)}.$$

$M(\omega)$ is the **magnitude** and $\sigma(\omega)$ is the **angle**. It is customary to express $M(\omega)$ in decibels (abbreviated by dB). Thus

$$M(\omega)|_{dB} = 20 \log_{10} M(\omega)$$

The angle is measured in degrees.

The **Singular Value Plot** is the plot of singular values of $H(j\omega)$ as a function of the frequency ω . If the system is a SISO system, the singular value plot is the same as the Bode diagram. The singular value plot is a useful tool in robustness analysis.

MATLAB Note: MATLAB functions **bode** and **sigma** can be used to draw the Bode plot and the singular value plot, respectively. For the Bode plot of MIMO system, the system is treated as arrays of SISO systems and the magnitudes and phases are computed for each SISO entry h_{ij} independently.

MATLAB function **freqresp** can be used to compute frequency response at same particular given frequencies or over grid of frequencies. When numerically safe, the frequency response is computed by diagonalizing the matrix A ; otherwise, Algorithm 5.5.1 is used.

5.6 Some Selected Software

5.6.1 MATLAB CONTROL SYSTEM TOOLBOX

Time response

- step - Step response.
- impulse - Impulse response.
- initial - Response of state-space system with given initial state.
- lsim - Response to arbitrary inputs.
- ltiview - Response analysis GUI.
- gensig - Generate input signal for LSIM.
- stepfun - Generate unit-step input.

Frequency response.

- bode - Bode plot for the frequency response.
- sigma - Singular value frequency plot.
- nyquist - Nyquist plot.
- nichols - Nichols chart.
- ltiview - Response analysis GUI.
- evalfr - Evaluate frequency response at given frequency.
- freqresp - Frequency response over a frequency grid.
- margin - Gain and phase margins.

5.6.2 MATCONTROL

- EXPMPADE - The Padé approximation to the exponential of a matrix
- EXPMSCHR - Computing the exponential of a matrix using Schur decomposition
- EXMPHESS - Computing the exponential of a matrix using Hessenberg decomposition
- FREQRESH - Computing the frequency response matrix using Hessenberg decomposition
- INTMEXP - Computing an integral involving a matrix exponentials

5.6.3 SLICOT

- MB05MD Matrix exponential for a real non-defective matrix
- MB05ND Matrix exponential and integral for a real matrix
- MB05OD Matrix exponential for a real matrix, with accuracy estimate

State-Space to Rational Matrix Conversion

TB04AD Transfer matrix of a state-space representation

State-Space to Frequency Response

TB05AD Frequency response matrix of a state-space representation

TF - Time Response

- TF01MD Output response of a linear discrete-time system
- TF01ND Output response of a linear discrete-time system (Hessenberg matrix)

5.6.4 MATRIX_X

Purpose: Gain and phase plots of discrete time systems.

Syntax:

[GAIN, DB, PHASE]=DBODE (SD, NS, OMEGANMIN, OMEGANMAX, NPTS, 'OPT') OR
 [GAIN, DB, PHASE]=DBODE (DNUM, DDEN, OMEGANMIN, OMEGANMAX, NPTS) OR
 [GIAN, DB, PHASE]=DBODE (SD, NS, OMEGAN)

Purpose: Initial value response of discrete time dynamic system.

Syntax: [N, Y]=DINITIAL (SD, NS, XO, NPTS)

Purpose: Step response of discrete time system.

Syntax: [N, Y]=DSTEP (SD, NS, NPTS) OR
 [N, Y]=DSTEP (DNUM, DDEN, NPTS)

Purpose: Frequency response of dynamic system. FREQ transforms the A matrix to Hessenberg form prior to finding the frequency response.

Syntax: [OMEGA, H]=FREQ (S, NS, RANGE, option) OR
 H=FREQ (S, NS, OMEGA, 'options')

Purpose: Compute the impulse response of a linear continuous time system.

Syntax: [T, Y]=IMPULSE (S, NS, TMAX, NPTS) OR
 [T, Y]=IMPULSE (NUM, DEN, TMAX, NPTS)

Purpose: Initial value response of continuous time dynamic system.

Syntax: [T, Y]=INITIAL (S, NS, XO, TMAX, NPTS)

Purpose: Response of continuous-time system to general inputs.

Syntax: [T, Y]=LSIM (S, NS, U, DELTAT, X0)

Purpose: Step response of continuous time system.

Syntax: [T, Y]=STEP (S, NS, TMAX, NPTS) OR
[T, Y]=STEP (NUM, DEN, TMAX, NPTS)

Purpose: Gives transfer function form of a state space system.

Syntax: [NUM, DEN]=TFORM (S, NS)

Purpose: Impulse response of continuous time system.

Syntax: [T, Y]=TIMR (S, NS, RANGE, 'MODE')

5.7 Summary and Review

I. State-Space Representations

A **continuous-time** linear time-invariant control system may be represented as:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t),\end{aligned}$$

where

- $x(t)$ is the **state vector**
- $u(t)$ is the **input vector**
- $y(t)$ is the **output vector**.

The matrices A, B, C and D are **time-invariant matrices**, known, respectively, as the **state matrix**, the **input matrix**, the **output matrix**, and the **direct transmission matrix**.

A **discrete-time** linear time-invariant control system may be represented as

$$\begin{aligned}x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t).\end{aligned}$$

where $x(t), u(t), y(t)$; and A, B, C and D have the same meaning as above.

II. Solutions of the Dynamical Equations

The solutions of the equations representing the continuous-time system in state-space form are given by (assuming $t_0 = 0$):

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-s)}Bu(s)ds$$

and

$$y(t) = Ce^{At}x_0 + \int_0^t Ce^{A(t-s)}Bu(s)ds + Du(t).$$

where x_0 is the value of $x(t)$ at $t = 0$. The matrix e^{At} is the **state-transition matrix in this case**.

The solutions of the equations representing the discrete-time system are given by

$$x(k) = A^k x(0) + \sum_{i=0}^{k-1} A^{k-1-i} Bu(i)$$

and

$$y(k) = CA^k x(0) + \left\{ \sum_{i=0}^{k-1} CA^{k-i-1} Bu(i) \right\} + Du(k)$$

III. Computing e^{At} .

There exist several methods for computing the state-transition matrix e^{At} . These include:

The Taylor series method, the Padé approximation method, ordinary differential equations methods, the eigenvalue-eigenvector method, and the matrix decomposition methods.

Among these, the **Padé approximation method with scaling and squaring**, (**Algorithm 5.3.1**), and the **method based on the Schur decomposition of A** (**Algorithm 5.3.2**) are the ones that are recommended for use in practice. If the problem is ill-conditioned, these methods, however, might not give accurate results, but, they are reliable. **ODE methods (Section 5.3.3) should be preferred if A is large and sparse.**

IV. Computing Integrals Involving Matrix Exponentials

An algorithm (**Algorithm 5.3.3**) is presented for computing integrals involving matrix exponentials.

V. Transfer Function Matrix

If $\hat{u}(s)$ and $\hat{y}(s)$ are the Laplace transforms of $u(t)$ and $y(t)$, then assuming zero initial condition we obtain

$$\hat{y}(s) = G(s)\hat{u}(s),$$

where

$$G(s) = C(sI - A)^{-1}B + D.$$

The matrix $G(s)$ is called transfer function matrix and is conveniently written as

$$G(s) \equiv \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right].$$

VI. The Frequency Response Matrix

The matrix $G(j\omega) = C(j\omega I - A)^{-1}B + D$ is called the **frequency response matrix**.

The frequency response plot for different values of ω is important in the study of different responses of a control system. For this the frequency response matrix needs to be computed.

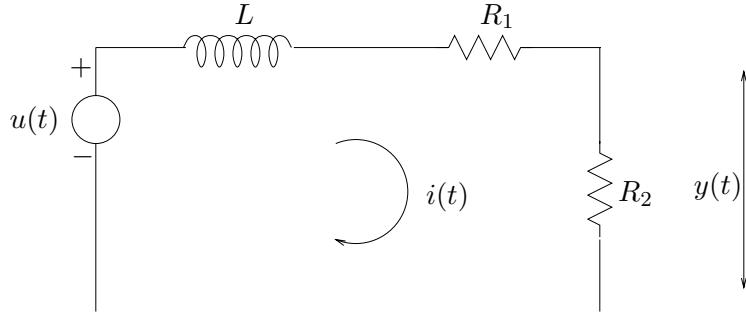
An efficient method (**Algorithm 5.5.1**), based on transformation of A to Hessenberg form, is described. **The Hessenberg method is nowadays widely used in practice.**

5.8 Chapter Notes and Further Reading

The examples on state-space model in Section 5.2.1 have been taken from various sources. These include the books by Brogan (1991), Chen (1984), Kailath (1980), Luenberger (1979), Szidarovszky and Bahill (1991) and the lecture notes by Sayed (1994), and Zhou and Doyle (1998). Discussions on system responses can be found in any standard text books. The books mentioned above and also the books by Brogan (1991), DeCarlo (1989), Dorf (1989), Friedland (1986), Patel and Munro (1982), etc., can be consulted. For various ways of computing the exponential matrix e^{At} , the paper by Moler and Van Loan (1978) is an excellent source. Some computational aspects of the exponential matrix have also been discussed in DeCarlo (1989). The frequency response algorithm is due to Laub (1981). For discussions on applications of frequency response matrix, see Safonov et al. (1981). Alternative algorithms for frequency response computation, see Misra and Patel (1988), Kenney et al (1993). The algorithm for computing integrals (**Algorithm 5.3.3**) involving matrix exponential has been taken from the paper of Van Loan (1978). The sensitivity analysis of the matrix e^{At} is due to Van Loan (1977). See also Golub and Van Loan (1996).

EXERCISES

1. (a) Consider the electric circuit

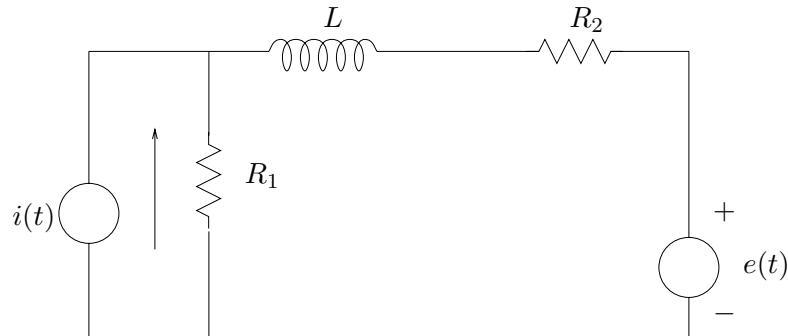


i. Show that the state-space representation of this circuit is given by

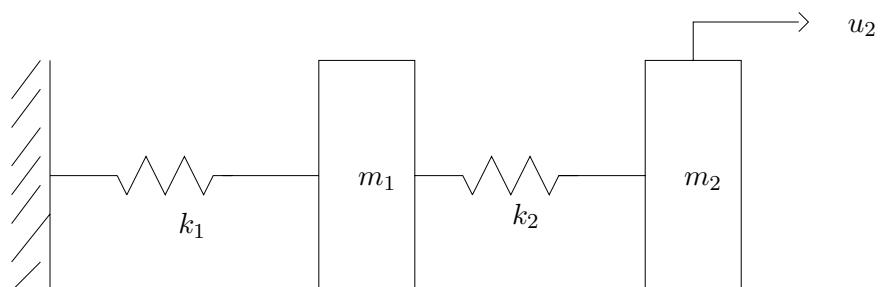
$$L \frac{di(t)}{dt} + (R_1 + R_2)i(t) = u(t), \quad y(t) = R_2i(t)$$

ii. Give an explicit expression for the solution of the state equation.

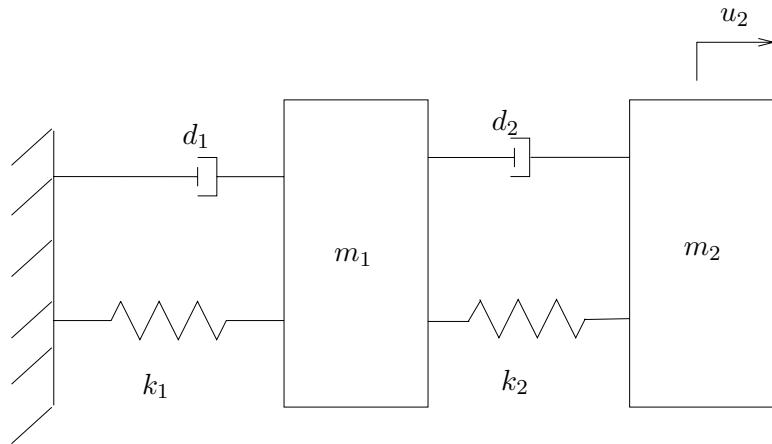
- (b) Write the state equations for the following electric network:



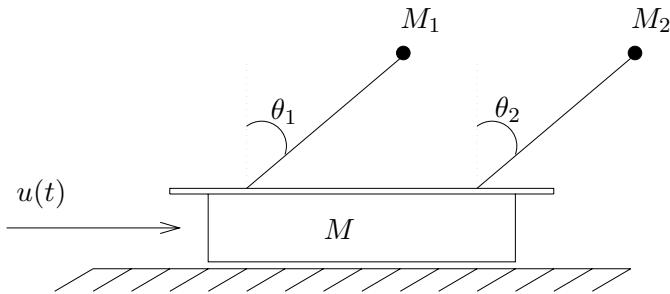
2. (a) Write down the equations of motion of the following spring-mass system in state-space form



- (b) Write down the state-space representation of the following spring-mass system:



3. Consider the following diagram of a cart of mass M carrying two sticks of masses M_1 and M_2 , and lengths l_1 , and l_2 .



- (a) Write down the equations of motion, in terms of the velocity v of the cart, u and $\theta_1, \theta_2, \dot{\theta}_1, \ddot{\theta}_2$.
- (b) Assuming θ_1 and θ_2 are small, linearize the equations of motion and then write down a first-order state-space representation.
4. (Saad(1988)) Consider the differential equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \beta \frac{\partial u}{\partial x} + \nu u + F(x, y, t)$$

on the unit square $\Omega = (0, 1) \times (0, 1)$, that models a chemical reaction process, where u represents the concentration of a chemical component that diffuses and convects. Let the boundary conditions $u(x, y, t) = 0$ for every t . Assume that $F(x, y, t)$ has the form

$$F(x, y, t) = F(x, y)g(t)$$

The term νu simulates a chemical reaction that results in an increase of the concentration that is proportional to u .

- (a) Discretize the region with n interior points in the x -direction and m interior points in the y -direction and show that this leads to the state-space representation of the form

$$\dot{u} = Au + bg,$$

where the dimension of A is nm .

- (b) Solve the above problem on a computer with $\beta = 20, \nu = 180, n = 10, m = 2$.

5. **(Lanchester War Model)** The following simple model of Warfare was developed by F. Lanchester in 1916.

Let x_1 and x_2 be the number of units in the two forces which are engaged in a war. Suppose that each unit of the first force has the “hitting” power h_1 and that of the second force has the “hitting” power of h_2 .

The “hitting” power is defined to be the number of casualties per unit time that one member can inflict on the other.

Suppose further that the hitting power of each force is directed uniformly against all units of the other force.

- (a) Develop a state-space representation of the model.

- (b) Show that

$$\begin{aligned}x_1(t) &= c_1 e^{t\sqrt{h_1 h_2}} + c_2 e^{-t\sqrt{h_1 h_2}} \\x_2(t) &= -c_1 \sqrt{\frac{h_1}{h_2}} e^{t\sqrt{h_1 h_2}} + c_2 \sqrt{\frac{h_1}{h_2}} e^{-t\sqrt{h_1 h_2}}\end{aligned}$$

where c_1 and c_2 are constants to be determined from initial conditions.

6. Suppose in a community there are only four classes ($i = 1, 2, 3, 4$). Make the following assumptions:

1. The number of men in each class in each generation is the same as the number of women.
2. Each person can marry only once and marries somebody in the same generation.
3. Each couple has exactly one son.

Let $x_i^{(k)}$ denote the number of men in class i in generation k . Suppose the following rules are adopted for producing male children in the community.

Rule for class 1. A son in class 1 is produced by every mother of this class and in no other way.

Rule for class 2. A son in this class is produced by every father in class 1 and by every mother in class 2 and in no other way.

Rule for class 3. A son in this class is produced by every father and every mother in class 3 and in no other way.

Rule for class 4. The number of sons in this class is equal to the total number of fathers in this class minus the number of the fathers who are married to class 1 or class 2.

- (a) Develop state-space equations for this model.
- (b) Find an expression for population at any generation k in terms of the initial populations.
- (c) What conclusions can you draw about the behavior of the social system of the community from the expression in (b)?

In particular, verify that the total population of the community is constant from generation to generation.

(For further details on this model, see Luenberger ((1979), pp. 105-108)).

7. (a) Find an explicit expression for the solution of the initial value problem

$$\dot{x}(t) = \begin{pmatrix} 0 & \lambda \\ -\lambda & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad x(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

- (b) Find the free response of the system.

8. Find an explicit expression for the solution of the homogeneous discrete-time system

$$x(t+1) = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} x(t), \quad x(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

9. Prove the following properties of e^{At} :

- (a) $e^{A(t+s)} = e^{At} \cdot e^{As}$
- (b) $e^{(A+B)t} = e^{At} \cdot e^{Bt}$ if and only if A and B commute.
- (c) e^{At} is nonsingular, and $(e^{At})^{-1} = e^{-At}$.
- (d) $\left(e^{\frac{A}{m}}\right)^m = e^A$, where m is an integer.
- (e) $e^{P^{-1}APt} = P^{-1}e^{At}P$.

10. Prove that the infinite series

$$e^{At} = \sum_{k=0}^{\infty} \frac{1}{k!} A^k t^k$$

converges uniformly and absolutely for t in any bounded interval.

11. Prove that $\frac{d}{dt}(e^{At}) = Ae^{At}$ (Hint: Differentiate $e^{At} = \sum_{k=0}^{\infty} \frac{1}{k!} A^k t^k$ term by term).

12. Illustrate the difficulty with the eigenvalue-eigenvector method for computing e^{At} with the matrix

$$A = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}$$

by choosing λ, μ , and α appropriately.

13. Let $R = (r_{ij})$ be an unreduced lower Hessenberg matrix and let $e^R = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_r \end{pmatrix}$, and

$B_i = R - r_{ii}I$. Then prove that

$$f_2 = \frac{1}{r_{12}} f_1 B_1$$

and

$$f_{i+1} = \frac{1}{r_{i-1,i}} (f_i B_i - \sum_{j=1}^{i-1} r_{ij} f_j), \quad i = 2, 3, \dots, n-1.$$

(Consult Datta and Datta (1976), if necessary). What difficulties do you foresee in computing e^R using these formulas? Give an illustrative example.

14. Compute e^A for

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

using

- (a) A fifth-order variable-step Runge-Kutta method.
- (b) The Adams-Moulton predictor correct formulas of variable order and variable step.
- (c) A general purpose ODE solver. (**Use error tolerance 10^{-6}**).

Compare the result in each case with that obtained by MATLAB function **expm**.

15. Let $X^{-1}AX = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$.

Then prove that

$$\|fl(e^{At}) - e^{At}\|_2 \leq n\mu e^{\rho(A)t} \cdot \text{Cond}_2(X).$$

16. (a) Write an algorithm, based on the block diagonal decomposition of A , to compute e^{At} :

$$A = X \operatorname{diag} (T_1, T_2, \dots, T_p) X^{-1}$$

- (b) Determine the flop-count of this algorithm.
(c) What numerical difficulty do you expect out of this algorithm?

17. Let

$$C = \begin{pmatrix} A & B \\ 0_{m \times n} & 0_{m \times m} \end{pmatrix}.$$

Then prove that $e^{Ct} = \begin{bmatrix} e^{At} & (\int_0^t e^{A(t-s)} ds)B \\ 0_{m \times n} & I_{m \times m} \end{bmatrix}$.

18. Prove that the matrix $D_{pq}(A)$ in the Padé approximation method is nonsingular if all the eigenvalues of A have negative real parts or if p and q are sufficiently large.

19. Develop an algorithm to compute A^s , where s is a positive integer, and A is an unreduced lower Hessenberg matrix. Apply your algorithm to compute A^{10} where $A =$

$$(i) \begin{pmatrix} 1 & 2 & 0 \\ 1 & 1 & 0.0001 \\ 1 & 1 & 1 \end{pmatrix} \text{ (ii) } \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \text{ (iii) } \begin{pmatrix} 1 & 10^{-3} & 0 \\ 1 & 1 & 10^{-4} \\ 1 & 1 & 1 \end{pmatrix}.$$

(Consult Datta and Datta (1976)).

20. Develop an algorithm to compute A^s , where A is an upper real Schur matrix, and s is a

positive integer. Apply your algorithm to compute A^5 , where $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 0.99 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1.99 \end{pmatrix}$.

21. Let $A = X\Lambda X^{-1}$, where $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$. Prove that $A^k = X\Lambda^k X^{-1}$, for each k . Under what conditions on $\lambda_1, \dots, \lambda_n$ does the infinite series of matrices $\sum c_k A^k$ converge?

22. Prove that the Laplace transform of $y(t) = e^{At}$ is $y(s) = (sI - A)^{-1}$.

23. Modify Algorithm 5.3.2 to compute e^A , where A is in real-Schur form.

24. Solve the initial value problem

$$\dot{x}(t) = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix} x(t) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u(t), \quad x(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

by using Laplace transform.

25. Show that $H(s) = C(sI - A)^{-1}B$ is the Laplace transform of the matrix $Ce^{At}B$.

26. Show that the transformation $\tilde{x} = Tx$, where T is nonsingular, preserves the transfer function.

27. Show that the transfer function of the system

$$\dot{x}(t) = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u, \quad x(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

$$y = (1, 1)x.$$

is $H(s) = \frac{s + \omega}{s^2 + \omega^2}$.

28. Applying z -transform to

$$x(t+1) = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} x(t), \quad x(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

show that $X(z) = \frac{z}{z^2 - z - 1}$. Hence solve the above initial value problem.

29. Modify the Hessenberg algorithm (Algorithm 5.5.1) for computation of the frequency response matrix that uses only real arithmetic. Give a flop-count of this modified algorithm.
30. Develop an algorithm for computing the frequency response matrix using formula (5.5.10) and the fact that the determinant of a matrix A is just the product of the diagonal entries of the matrix U in its LU factorization. (Consult Misra and Patel (1988)).
31. Develop an algorithm for computing the frequency response matrix, based on the reduction of A to real Schur form. Compare the flop-count of this algorithm with that of the Hessenberg algorithm.
32. Develop an algorithm for computing the frequency response matrix of the descriptor model

$$E\dot{x} = Ax + Bu$$

based on the initial reduction of the pair (A, B) to Hessenberg-triangular form described in Chapter 4.

References

1. C.T. Baldwin, *Fundamentals of Electrical Measurements*, George G. Harp and Co. Ltd., London, 1961.
2. W.L. Brogan, *Modern Control Theory*, 3rd Edition, Prentice Hall, Englewood Cliffs, NJ, 1991.
3. S.L. Campbell, *Singular Systems of Differential Equations*, Pitman, Marshfield, MA, 1980.
4. C.-T. Chen, *Linear System Theory and Design*, CBS College Publishing, New York, 1984.
5. B.N. Datta and K. Datta, An algorithm to compute the powers of a Hessenberg matrix and applications, *Lin. Alg. Appl.* vol. 14, pp. 273-284, 1976.
6. R.A. DeCarlo, *Linear Systems: A State Variable Approach with Numerical Implementation*, Prentice Hall, Englewood Cliff, NJ, 1989.
7. R.C. Dorf, and R.H. Bishop, *Modern Control Systems*, Prentice Hall, Upper Saddle River, NJ, 2001.
8. G.F. Franklin, J.D. Powell, and M.L. Workman, *Digital Control of Dynamic Systems*, 2nd Edition, Addison-Wesley Publishing Company, New York, 1990.
9. B. Friedland, *Control Systems Design: An Introduction to State-Space Methods*, McGraw-Hill, New York, 1986.
10. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.
11. T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.
12. C.S. Kenney, A.J. Laub and S.C. Stubberud, Frequency response computation via rational interpolation, *IEEE Trans. Automat. Control*, vol. 38, pp.1203-1213, 1993.
13. A.J. Laub, Efficient multivariable frequency response computations, *IEEE Trans. Automat. Control*, vol. AC-26, pp. 407-408, 1981.
14. A.J. Laub and A. Linnemann, Hessenberg and Hessenberg/triangular forms in linear systems theory, *Int. J. Control*, vol. 44, pp. 1523-1547, 1986.
15. D.G. Luenberger, *Introduction to Dynamic Systems: Theory, Methods, & Applications*, John Wiley & Sons, New York, 1979.
16. P. Misra, Hessenberg-triangular reduction and transfer function matrices of singular systems, *IEEE Trans. Circuits Systems*, vol. CAS-36, pp. 907-912, 1989.

17. P. Misra and R.V. Patel, A determinant identity and its application in evaluating frequency response matrices, *SIAM J. Matrix Anal. Appl.*, vol. 9, pp. 248-255, 1988.
18. C.B. Moler and C.F. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, *SIAM Review*, vol. 20, pp. 801-836, 1978.
19. B.N. Parlett, A recurrence among the elements of functions of triangular matrices, *Lin. Alg. Appl.*, vol. 29, pp. 323-346, 1976.
20. R.V. Patel and N. Munro, *Multivariable Systems Theory and Design*, Pergamon Press, Oxford, UK, 1982.
21. M.G. Safonov, A.J. Laub, and G.L. Hartman, Feedback properties of multivariable systems: The role and use of the return difference matrix, *IEEE Trans. Automat. Contr.*, vol. AC-26, pp. 47-65, 1981.
22. Ali Sayed, *Lecture Notes on Dynamical Systems*, University of California, Los Angeles, 1994.
23. T.T. Soong, *Active Structural Control: Theory and Practice*, Longman Scientific and Technical, Essex, UK, 1990.
24. F. Szidarovszky and A. Terry Bahill, *Linear Systems Theory*, CRC Press, Boca Raton, 1991.
25. C.F. Van Loan, The Sensitivity of the matrix exponential, *SIAM J. Numer. Anal.*, vol. 14, pp. 971-981, 1977.
26. C.F. Van Loan, Computing integrals involving the matrix exponential, *IEEE Trans. Automat. Control*, vol. AC-23, pp. 395-404, 1978.
27. K. Zhou (with J. Doyle), *Essentials of Robust Control*, Prentice Hall, Upper Saddle River, NJ, 1998.

Chapter 6

CONTROLLABILITY, OBSERVABILITY AND DISTANCE TO UNCONTROLLABILITY

Contents

6.1	Introduction	192
6.2	Controllability: Definitions and Basic Results	193
6.2.1	Controllability of a Continuous-Time System	193
6.2.2	Controllability of a Discrete-Time System	197
6.3	Observability: Definitions and Basic Results	198
6.3.1	Observability of a Continuous-time System	198
6.3.2	Observability of a Discrete-Time System	200
6.4	Decompositions of Uncontrollable and Unobservable Systems.	200
6.5	Controller and Observer Canonical Forms	202
6.6	Numerical Difficulties with Theoretical Criteria of Controllability and Observability	204
6.7	A Numerically Effective Test of Controllability	207
6.8	A Numerically Effective Test of Observability	216
6.9	Distance to an Uncontrollable System	216
6.9.1	Newton's and the Bisection Methods for Computing the Distance to Uncontrollability	218
6.9.2	The Wicks-DeCarlo Method for Distance to Uncontrollability	222
6.9.3	A Global Minimum Search Algorithm.	224
6.10	Distance to Uncontrollability and the Singular Values of the Controllability Matrix	224

6.11 Some Selected Software	225
6.11.1 MATLAB CONTROL SYSTEM TOOLBOX	225
6.11.2 MATCONTROL	225
6.11.3 CSP-ANM	225
6.11.4 SLICOT	226
6.11.5 MATRIX _X	226
6.12 Summary and Review	226
6.13 Chapter Notes and Further Reading	228

Topics Covered

- Controllability and Observability Criteria
- Controller and Observer Companion Forms.
- Kalman Decomposition
- Controllable and Observable Realizations of the Transfer Matrix
- Controller and Observer Hessenberg Forms
- Distance to Uncontrollability

6.1 Introduction

This Chapter deals with discussions on the two most fundamental notions, **controllability** and **observability**, and related concepts. The well-known criteria of controllability and observability, such as the **Kalman criterion** and **those based on eigenvalues and eigenvectors**, are stated and proved in Theorem 6.2.1.

These theoretically important criteria, unfortunately, do not yield numerically effective tests of controllability. This is demonstrated by means of some examples and discussions in **Section 6.6. Numerically effective tests**, based on reduction of the pairs (A, B) and (A, C) , respectively, to the **controller-Hessenberg** and **observer-Hessenberg** pairs, achieved by means of orthogonal similarly, are described in **Section 6.7 and 6.8**.

Controllability and observability are generic concepts. What is more important in practice is to know when a controllable system is close to an uncontrollable one. To this end, a measure of the distance to uncontrollability is introduced in **Section 6.9** and a characterization (**Theorem 6.9.1**) in terms of the minimum singular value of a certain matrix is stated and proved. Finally, two algorithms (**Algorithm 6.9.1** and **Algorithm 6.9.2**) are described to measure the distance to uncontrollability.

The Chapter concludes with a brief discussion (**Section 6.10**) on the relationship between the distance to uncontrollability and the singular values of the controllability matrix. The important message here is that the singular values of the controllability matrix as such cannot be used to make a prediction of how close the system is to an uncontrollable system. It is the largest gap between two singular values that should be considered.

Reader's Guide for Chapter 6

The readers having knowledge of basic concepts and results on controllability and observability, can skip sections 6.2-6.5.

6.2 Controllability: Definitions and Basic Results

In this section, we introduce the basic concepts and some algebraic criteria of controllability.

6.2.1 Controllability of a Continuous-Time System

Definition 6.2.1 *The system*

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}\tag{6.2.1}$$

is said to be **controllable**, if starting from any initial state $x(0)$, the system can be driven to any final state $x_1 = x(t_1)$ in some finite time t_1 , choosing the input variable $u(t), 0 \leq t \leq t_1$ appropriately.

Remark: The controllability of the system (6.2.1) is often referred to as the controllability of the pair (A, B) ; the reason for which will be clear in the following theorem.

Theorem 6.2.1 (Criteria for Continuous-time Controllability) Let $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ ($m \leq n$).

The following are equivalent:

(i) The system (6.2.1) is controllable.

(ii) The $n \times nm$ matrix

$$C_M = (B, AB, A^2B, \dots, A^{n-1}B)$$

has full rank n (**The Kalman Criterion of Controllability**)

(iii) The matrix

$$W_C = \int_0^{t_1} e^{At} BB^T e^{A^T t} dt$$

is nonsingular for any $t_1 > 0$.

(iv) If (λ, x) is an eigenpair of A^T , i.e., $x^T A = \lambda x^T$, then $x^T B \neq 0$

(v) Rank $(A - \lambda I, B) = n$ for every eigenvalue λ of A .

(vi) The eigenvalues of $A - BK$ can be arbitrarily assigned (assuming that the complex eigenvalues occur in conjugate pairs) by a suitable choice of K .

Proof: Without loss of generality, we can assume that $t_0 = 0$. Let $x(0) = x_0$.

(i) \Rightarrow (ii). Suppose that the rank of C_M is not n . From Chapter 5, we know that

$$x(t_1) = e^{At_1} x_0 + \int_0^{t_1} e^{A(t_1-t)} Bu(t) dt. \quad (6.2.2)$$

That is,

$$\begin{aligned} x(t_1) - e^{At_1} x_0 &= \int_0^{t_1} \left\{ I + A(t_1-t) + \frac{A^2}{2!}(t_1-t)^2 + \dots \right\} Bu(t) dt \\ &= B \int_0^{t_1} u(t) dt + AB \int_0^{t_1} (t_1-t)u(t) dt + A^2 B \int_0^{t_1} \frac{(t_1-t)^2}{2!} u(t) dt + \dots \end{aligned}$$

From the Cayley-Hamilton Theorem (see Chapter 1), it then follows that the vector $x(t_1)$ is a linear combination of the columns of $B, AB, A^2B, \dots, A^{n-1}B$.

Since C_M does not have rank n , then these columns vectors can not form a basis of the state-space and therefore for some t_1 , $x(t_1) = x_1$ can not be attained, implying that (6.2.1) is not controllable.

(ii) \Rightarrow (iii). Suppose that the matrix C_M has rank n , but the matrix

$$W_C = \int_0^{t_1} e^{At} BB^T e^{A^T t} dt \quad (6.2.3)$$

is singular.

Let v be a nonzero vector that $W_C v = 0$. Then $v^T W_C v = 0$. That is,

$\int_0^{t_1} v^T e^{At} BB^T e^{A^T t} v dt = 0$. The integrand is always nonnegative, since it is of the form $c^T(t)c(t)$, where $c(t) = B^T e^{A^T t} v$. Thus for the above integral to be equal to zero, we must have

$$v^T e^{At} B = 0, \text{ for } 0 \leq t \leq t_1.$$

From this we obtain (evaluating the successive derivatives with respect t , at $t = 0$):

$$v^T A^i B = 0, \quad i = 1, 2, \dots, n - 1.$$

That is, v is orthogonal to all columns of the matrix C_M . Since it has been assumed that the matrix C_M has rank n , this means that $v = 0$, which is a contradiction.

(iii) \Rightarrow (i). We show that $x(t_1) = x_1$. Let us now choose the vector $u(t)$ as

$$u(t) = B^T e^{A^T(t_1-t)} W_C^{-1} (-e^{At_1} x_0 + x_1).$$

Then from (6.2.2), it is easy to see that

$$x(t_1) = x_1.$$

This implies that the system (6.2.1) is controllable.

(ii) \Rightarrow (iv). Let x be an eigenvector of A^T corresponding to an eigenvalue λ ; that is

$$x^T A = \lambda x^T.$$

Suppose that $x^T B = 0$. Then

$$x^T C_M = (x^T B, \lambda x^T B, \lambda^2 x^T B, \dots, \lambda^{n-1} x^T B) = 0.$$

Since the matrix C_M has full rank, $x = 0$, which is a contradiction.

(iv) \Rightarrow (ii). Assume that none of the eigenvectors of A^T is orthogonal to the columns of B , but $\text{rank}(C_M) = k < n$.

We will see later in this chapter (Theorem 6.4.1) that, in this case, there exists a nonsingular matrix T such that

$$\bar{A} = T A T^{-1} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & \bar{A}_{22} \end{pmatrix}, \quad \bar{B} = T B = \begin{pmatrix} \bar{B}_1 \\ 0 \end{pmatrix}, \quad (6.2.4)$$

where \bar{A}_{22} is of order $(n - k)$, and $k = \text{rank}(C_M)$.

Let v_2 be an eigenvector of $(\bar{A}_{22})^T$ corresponding to an eigenvalue λ .

Then

$$(\bar{A})^T \begin{pmatrix} 0 \\ v_2 \end{pmatrix} = \begin{pmatrix} \bar{A}_{11}^T & 0 \\ \bar{A}_{12}^T & \bar{A}_{22}^T \end{pmatrix} \begin{pmatrix} 0 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 \\ \bar{A}_{22}^T v_2 \end{pmatrix} = \lambda \begin{pmatrix} 0 \\ v_2 \end{pmatrix}.$$

Furthermore,

$$(0, v_2^T) \bar{B} = (0, v_2^T) \begin{pmatrix} \bar{B}_1 \\ 0 \end{pmatrix} = 0.$$

That is, there is an eigenvector, namely $\begin{pmatrix} 0 \\ v_2 \end{pmatrix}$, of $(\bar{A})^T$ such that it is orthogonal to the columns of \bar{B} . This means that the pair (\bar{A}, \bar{B}) , is not controllable.

This is a contradiction because a similarity transformation does not change controllability.

(ii) \Rightarrow (v). Rank $(\lambda I - A, B) < n$ if and only if there exists a nonzero vector v such that $v^T (\lambda I - A, B) = 0$.

This equation is equivalent to

$$A^T v = \lambda v, \text{ and } v^T B = 0.$$

This means that v is an eigenvector of A^T corresponding to the eigenvalue λ and it is orthogonal to the columns of B . The system (A, B) is, therefore, not controllable, by (iv).

(v) \Rightarrow (ii). If (v) were not true, then from (iv), we would have had

$$x^T (B, AB, \dots, A^{n-1} B) = 0,$$

meaning that rank (C_M) is less than n .

(vi) \Rightarrow (i). Suppose that (vi) holds, but not (i). Then the system can be decomposed in the form (6.2.4) such that a subsystem corresponding to \bar{A}_{22} is uncontrollable, whose eigenvalues, therefore, cannot be changed by the control. This contradicts (vi).

(i) \rightarrow (vi). The proof of this part will be given in Chapter 10 (**Theorem 10.4.1**). It will be shown there that if (A, B) is controllable, then there exists a matrix K such that the eigenvalues of the matrix $(A - BK)$ are in desired locations.

Definition 6.2.2 *The matrix*

$$C_M = (B, AB, A^2 B, \dots, A^{n-1} B) \tag{6.2.5}$$

*is called the **controllability matrix**.*

Remarks: The eigenvector criterion (iv) and the eigenvalue criterion (v) are popularly known as the **Popov-Belevitch-Hautus (PBH)** criteria of controllability (see Hautus (1969)). For a historical perspective of this title, see Kailath (1980, 135).

Component Controllability. The controllability, as defined in the Definition 6.2.1, is often referred to as the **complete controllability** implying that all the states are controllable.

However, if only one input, say $u_j(t)$, from $u(t) = (u_1(t), \dots, u_m(t))^T$ is used, then the rank of the corresponding $n \times n$ controllability matrix $C_M^j = (b_j, Ab_j, \dots, A^{n-1}b_j)$, where b_j is the j -th column of B , determines the number of states that are controllable using the input $u_j(t)$. This is illustrated in the following example.

Consider Example 5.2.1 on the motions of an orbiting satellite, with $d_0 = 1$.

It is easy to see that C_M has rank 4, so that all states are controllable using both inputs. However, if only the first input $u_1(t)$ is used, then

$$C_M^1 = (b_1, Ab_1, A^2b_1, A^3b_1) = \begin{pmatrix} 0 & 1 & 0 & -\omega^2 \\ 1 & 0 & -\omega^2 & 0 \\ 0 & 0 & -2\omega & 0 \\ 0 & -2\omega & 0 & 2\omega^3 \end{pmatrix}$$

which is singular.

Thus one of the states is not controllable by using only the radial force $u_1(t)$.

However, it can be easily verified that all the states would be controllable if the tangential force $u_2(t)$ were used instead of $u_1(t)$.

Controllable and Uncontrollable Modes

From the eigenvalue and eigenvector criteria above, it is clear that the controllability and uncontrollability of the pair (A, B) are tied with the eigenvalues and eigenvectors of the system matrix A .

Definition 6.2.3 *A mode of the system (6.2.1) or, equivalently, an eigenvalue λ of A is controllable if the associated left eigenvector (that is the eigenvector of A^T associated with λ) is not orthogonal to the columns of B . Otherwise, the mode is uncontrollable. ■*

6.2.2 Controllability of a Discrete-Time System

Definition 6.2.4 *The discrete-time system*

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{6.2.6}$$

is said to be controllable if for any two states x_0 and x_1 , there exists a finite sequence of inputs $\{u_0, u_1, \dots, u_{N-1}\}$ that transfers x_0 to x_1 ; that is, $x(N) = x_1$.

*In particular, if $x_0 = 0$ and the system (6.2.6) is controllable, then it is called **reachable** (see Chen (1984)).*

Note. To avoid any confusion, we will assume (without any loss of generality) that $x_0 = 0$. So, in our case, the notion of controllability and reachability are equivalent.

Most of the criteria on controllability in the continuous-time case also hold in the discrete-time case. Here we will state and prove only one criterion analogous to (ii) of Theorem 6.2.1.

Theorem 6.2.2 *The discrete-time system (6.2.6) or the pair (A, B) is controllable if and only if the rank of the controllability matrix*

$$C_M = (B, AB, \dots, A^{n-1}B)$$

is n .

Proof: From Theorem 5.4.1, we know that the general solution of the discrete-time systems is

$$x_N = A^{N-1}Bu_0 + A^{N-2}Bu_1 + \dots + Bu_{N-1}$$

Thus, x_N can be expressed as a linear combination of $A^{k-1}B$, $k = N, \dots, 1$.

So, it is possible to choose u_0 through u_{N-1} for arbitrary x_N if and only if the sequence $\{A^iB\}$ has a finite number of columns that span \mathbb{R}^n ; and this is possible, if and only if the controllability matrix C_M has rank n ■.

6.3 Observability: Definitions and Basic Results

In this section we state definitions and basic results of observability. The results will not be proved here; because they can be easily proved by duality of the results on controllability proved in the previous section.

6.3.1 Observability of a Continuous-time System

The concept of observability is dual to the concept of controllability.

Definition 6.3.1 *The continuous-time system (6.2.1) is said to be **observable** if there exists $t_1 > 0$ such that the initial state $x(0)$ can be uniquely determined from the knowledge of $u(t)$ and $y(t)$ for all $t, 0 \leq t \leq t_1$.*

Remark: The observability of the system (6.2.1) is often referred to as the observability of the pair (A, C) .

Analogous to the case of controllability, we state the following criteria of observability.

Theorem 6.3.1 (Criteria for Continuous-Time Observability)

The following are equivalent:

- (i) The system (6.2.1) is observable.
- (ii) The observability matrix

$$O_M = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix}$$

has full rank n .

(iii) The matrix

$$W_O = \int_0^{t_1} e^{A^T \tau} C^T C e^{A\tau} d\tau$$

is nonsingular for any $t_1 > 0$.

(iv) The matrix $\begin{bmatrix} \lambda I - A \\ C \end{bmatrix}$ has rank n for every eigenvalue λ of A .

(v) None of the eigenvectors of A is orthogonal to the rows of C , that is, if (λ, y) is an eigenpair of A , then $Cy \neq 0$.

(vi) There exists a matrix L such that the eigenvalues of $A + LC$ can be assigned arbitrarily, provided that the complex eigenvalues occur in conjugate pairs.

We only prove (iii) \iff (i) and leave the others as an exercise [Exercise 6].

Theorem 6.3.2 *The pair (A, C) is observable if and only if the matrix W_O is nonsingular for any $t_1 > 0$.*

Proof: First suppose that the matrix W_O is nonsingular. Since $y(t)$ and $u(t)$ are known, we can assume, without any loss of generality, that $u(t) = 0$ for every t . Thus,

$$y(t) = C e^{At} x(0).$$

This gives

$$W_O x(0) = \int_0^{t_1} e^{A^T \tau} C^T C y(\tau) d\tau.$$

Thus $x(0)$ is uniquely determined and is given by $x(0) = W_O^{-1} \int_0^{t_1} e^{A^T \tau} C^T C y(\tau) d\tau$.

Conversely, if W_O is singular, then there exists a nonzero vector z such that $W_O z = 0$, which in turn implies that $C e^{A\tau} z = 0$. So, $y(\tau) = C e^{A\tau} (x(0) + z) = C e^{A\tau} x(0)$.

Thus $x(0)$ cannot be determined uniquely, implying that (A, C) is not observable. ■

Component Observability. As in the case of controllability, we can also speak of component observability when all the states are not observable by certain output. The rank of the observability matrix

$$C_M^j = \begin{pmatrix} c_j \\ c_j A \\ \vdots \\ c_j A^{n-1} \end{pmatrix},$$

where c_j is the j th row of the output matrix C , determines the number of states that are observable by the output $y_j(t)$.

6.3.2 Observability of a Discrete-Time System

Definition 6.3.2 *The discrete-time system (6.2.6) is said to be observable if there exists an index N such that the initial state x_0 can be completely determined from the knowledge of inputs $u_0, u_1 \dots u_{N-1}$, and the outputs y_0, y_1, \dots, y_N .*

The criteria of observability in the discrete-time case are the same as in the continuous-time case, and therefore, will not be repeated here.

6.4 Decompositions of Uncontrollable and Unobservable Systems.

Suppose that the pair (A, B) is not controllable. Let the rank of the controllability matrix be $k < n$. Then the following theorem shows that the system can be decomposed into controllable and uncontrollable parts.

Theorem 6.4.1 (Decomposition of Uncontrollable System). *If the controllability matrix C_M has rank k , then there exists a nonsingular matrix T such that*

$$\bar{A} = TAT^{-1} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & \bar{A}_{22} \end{pmatrix}, \quad \bar{B} = TB = \begin{pmatrix} \bar{B}_1 \\ 0 \end{pmatrix} \quad (6.4.1)$$

where \bar{A}_{11} , \bar{A}_{12} , and \bar{A}_{22} are, respectively, of order $k \times k$, $k \times (n - k)$, and $(n - k) \times (n - k)$, and \bar{B}_1 has k rows. Furthermore, $(\bar{A}_{11}, \bar{B}_1)$ is controllable.

Proof: Let v_1, \dots, v_k be the independent columns of the controllability matrix C_M . We can always choose a set of $n - k$ vectors v_{k+1}, \dots, v_n so that the vectors $(v_1, v_2, \dots, v_k, v_{k+1}, \dots, v_n)$ form a basis of \mathbb{R}^n .

Then it is easy to see that the matrix $T^{-1} = (v_1, \dots, v_n)$ is such that TAT^{-1} and TB will have the above forms.

To prove that $(\bar{A}_{11}, \bar{B}_1)$ is controllable, we note that the controllability matrix of the pair (\bar{A}, \bar{B}) is

$$\begin{pmatrix} \bar{B}_1 & \bar{A}_{11}\bar{B}_1 & \cdots & (\bar{A}_{11})^{k-1}\bar{B}_1 & \cdots & (\bar{A}_{11})^{n-1}\bar{B}_1 \\ 0 & 0 & & 0 & \cdots & 0 \end{pmatrix}$$

Since, for each $j \geq k$, $(\bar{A}_{11})^j$ is a linear combination of $(\bar{A}_{11})^i, i = 0, 1, \dots, (k-1)$, by the Cayley-Hamilton Theorem (see Chapter 1), we then have $\text{rank}(\bar{B}_1, \bar{A}_{11}\bar{B}_1, \dots, \bar{A}_{11}^{k-1}\bar{B}_1) = k$, proving that $(\bar{A}_{11}, \bar{B}_1)$ is controllable. ■

Note: If we define $\bar{x} = Tx$, then the state vector \bar{x} corresponding to the system defined by \bar{A} , and \bar{B} is given by $\bar{x} = \begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \end{pmatrix}$.

Remark (Choosing T orthogonal)

Note that T in Theorem 6.4.1 can be chosen to be orthogonal by finding the QR factorization of the controllability matrix C_M . Thus, if $C_M = QR$, then $T = Q^T$.

■

Using duality, we have the following decomposition for an unobservable pair. The proof is left as an exercise [Exercise 7].

Theorem 6.4.2 (Decomposition of Unobservable System). *If the observability matrix O_M has rank $k' < n$, then there exists a nonsingular matrix T such that*

$$\bar{A} = TAT^{-1} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & \bar{A}_{22} \end{pmatrix}, \bar{C} = CT^{-1} = (0, \bar{C}_1) \quad (6.4.2)$$

with $(\bar{A}_{11}, \bar{C}_1)$ observable and \bar{A}_{11} is of order k' . ■

The Kalman Decomposition

Combining the above two Theorems, we obtain the following decomposition, known as the **Kalman Canonical Decomposition**. The proof of the Theorem is left as an exercise [Exercise 8], and can also be found in any standard text on linear systems theory.

Theorem 6.4.3 (The Kalman Canonical Decomposition Theorem). *Given the system (6.2.1) there exists a nonsingular coordinate transformation $\bar{x} = Tx$ such that*

$$\begin{pmatrix} \dot{\bar{x}}_{c\bar{o}} \\ \dot{\bar{x}}_{co} \\ \dot{\bar{x}}_{\bar{c}\bar{o}} \\ \dot{\bar{x}}_{\bar{c}o} \end{pmatrix} = \begin{pmatrix} \bar{A}_{c\bar{o}} & \bar{A}_{12} & \bar{A}_{13} & \bar{A}_{14} \\ 0 & \bar{A}_{co} & 0 & \bar{A}_{24} \\ 0 & 0 & \bar{A}_{\bar{c}\bar{o}} & A_{34} \\ 0 & 0 & 0 & A_{\bar{c}o} \end{pmatrix} \begin{pmatrix} \bar{x}_{c\bar{o}} \\ \bar{x}_{co} \\ \bar{x}_{\bar{c}\bar{o}} \\ \bar{x}_{\bar{c}o} \end{pmatrix} + \begin{pmatrix} \bar{B}_{c\bar{o}} \\ \bar{B}_{co} \\ 0 \\ 0 \end{pmatrix} \quad (6.4.3)$$

$$y = (0, \bar{C}_{co}, 0, \bar{C}_{\bar{co}}) \begin{pmatrix} \bar{x}_{c\bar{o}} \\ \bar{x}_{co} \\ \bar{x}_{\bar{c}\bar{o}} \\ \bar{x}_{\bar{co}} \end{pmatrix} + Du$$

$\bar{x}_{c\bar{o}}$ *states which are controllable but not observable.*

\bar{x}_{co} *states which are both controllable and observable*

$\bar{x}_{\bar{c}\bar{o}}$ *states which are uncontrollable and unobservable*

$\bar{x}_{\bar{co}}$ *states which are uncontrollable but observable.*

Moreover, the transfer function matrix from u to y is given by

$$G(s) = \bar{C}_{co}(sI - \bar{A}_{co})^{-1}\bar{B}_{co} + D.$$

■

Remark: It is interesting to observe that the uncontrollable and/or unobservable parts of the system do not appear in the description of the transfer function matrix.

6.5 Controller and Observer Canonical Forms

An important property of a linear system is that **controllability and observability remains invariant under certain transformations**. We will state the result for controllability without proof. A similar result, of course, holds for observability. Proofs are left as [Exercises 9].

Theorem 6.5.1 *Let T be a nonsingular matrix such that $TAT^{-1} = \tilde{A}$, and $TB = \tilde{B}$, then (A, B) is controllable if and only if (\tilde{A}, \tilde{B}) is controllable.* ■

The question naturally arises if the matrix T can be chosen so that \tilde{A} and \tilde{B} will have simple forms, from where conclusions about controllability or observability can be easily drawn. Two such forms, **controller-canonical form** (or the **controller-companion form**) and the **Jordan Canonical form** are well-known in control text books (Kailath (1980), Chen (1984), Szidarovszky and Bahill (1991), Luenberger (1979), etc.). **Unfortunately, neither of these two forms, in general, can be obtained in a numerically stable way; because, T , not being an orthogonal matrix in general, can be highly ill-conditioned.**

The Controller and Observer Canonical forms are, however, very valuable theoretical tools in establishing many theoretical results in control and systems theory (e.g., the proof of the eigenvalue assignment theorem by state feedback (**Theorem 10.4.1 in Chapter 10**)). We just state these two forms here for later uses as theoretical tools. First consider the single-input case.

Let (A, b) be controllable and let C_M be the controllability matrix.

Let s_n be the last row of C_M^{-1} . Then the matrix T defined by

$$T = \begin{pmatrix} s_n \\ s_n A \\ \vdots \\ s_n A^{n-1} \end{pmatrix} \quad (6.5.1)$$

is such that

$$\tilde{A} = TAT^{-1} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & & 1 \\ -a_1 & -a_2 & -a_3 & \cdots & -a_n \end{pmatrix}, \quad \tilde{b} = Tb = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \quad (6.5.2)$$

Similarly, it is possible to show that, if (A, b) is controllable, then there exists a nonsingular matrix P such that

$$PAP^{-1} = \begin{pmatrix} -a_n & -a_{n-1} & \cdots & -a_2 & -a_1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}, \quad Pb = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \quad (6.5.3)$$

The forms (6.5.1) and (6.5.3) are, respectively, known as **lower and upper companion (or controller) canonical forms**. By duality, the **observer-canonical forms** (in lower and upper companion forms) can be defined. Thus, the pair (\tilde{A}, \tilde{c}) given by

$$\tilde{A} = \begin{pmatrix} 0 & 0 & \cdots & 0 & -a_1 \\ 1 & 0 & \cdots & 0 & -a_2 \\ 0 & 1 & \cdots & 0 & -a_3 \\ \vdots & & & & \vdots \\ 0 & 0 & \cdots & 1 & -a_n \end{pmatrix}, \quad \tilde{c} = (0, 0, \dots, 0, 1).$$

is an **upper observer canonical form**.

MATCONTROL Note: The MATCONTROL function `cntrlc` can be used to find a controller-canonical form.

The Luenberger-Canonical Form

In the multi-input case, the controllable pair (A, B) can be reduced to the pair (\tilde{A}, \tilde{B}) , where $\tilde{A} = TAT^{-1}$ is a block matrix, where each diagonal block matrix is a companion matrix of the form given in (6.5.2), and \tilde{B} has the form

$$\tilde{B} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \\ 1 & x & \dots & x \\ \hline 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & x \\ \hline & & \vdots & \\ \hline 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 1 \end{bmatrix} \quad (6.5.4)$$

The number of diagonal blocks of \tilde{A} is equal to the rank of B . “ x ” denotes a possible nonzero entry. Such a form is known as the **Luenberger controller-canonical form**. Similarly, by duality, the **Luenberger observer canonical form** can be written down.

Numerical Instability: In general, like a controller companion form, the Luenberger canonical form also cannot be obtained in a numerically stable way.

6.6 Numerical Difficulties with Theoretical Criteria of Controllability and Observability

Each of the algebraic criterion of controllability (observability) described in Section 6.2 (Section 6.3) suggests a test of controllability or (observability). Unfortunately, most of them do not lead to numerically viable tests as the following discussions show. First, let’s look into the Kalman criterion of controllability.

The Kalman criterion requires successive matrix multiplications and determination of the rank of an $n \times nm$ matrix.

It is well known that matrix multiplications involving nonorthogonal matrices can lead to a severe loss of accuracy (see Chapter 3). The process may transform the problem to a more sensitive one. To illustrate this, consider the following illuminating example from Paige (1981).

Example 6.6.1

$$A = \begin{pmatrix} 1 & & & \\ & 2^{-1} & & \\ & & \ddots & \\ & & & 2^{-9} \end{pmatrix}_{10 \times 10}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

The pair (A, B) is clearly controllable. The controllability matrix (B, AB, \dots, A^9B) can be computed easily and stored accurately. Note that the (i, j) -th entry of this matrix is $2^{(-i+1)(j-1)}$. This matrix has three smallest singular values $0.613 \times 10^{-12}, 0.364 \times 10^{-9}$ and 0.712×10^{-7} . Thus, on a computer with machine precision no smaller than 10^{-12} , one will conclude that the *numerical rank of this matrix* is less than 10, indicating that the system is uncontrollable. (Recall that matrix A is said to have a *numerical rank* r if the computed singular values $\tilde{\sigma}_i, i = 1, \dots, n$ satisfy $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \dots \geq \tilde{\sigma}_r \geq \delta \geq \tilde{\sigma}_{r+1} \geq \dots \geq \tilde{\sigma}_n$, where δ is a tolerance).

Note that determining the rank of a matrix using the singular values is the most effective way from a numerical viewpoint. ■

The criteria (iv)-(vi) in Theorem 6.2.1 are based on eigenvalues and eigenvectors computations. We know that the eigenvalues and eigenvectors of certain matrices can be very ill-conditioned and that the ill-conditioning of the computed eigenvalues will again lead to inaccuracies in computations. For example, by criteria (vi) of controllability in Theorem 6.2.1, it is possible, when (A, B) is controllable, to find a matrix K such that $A + BK$ and A have disjoint spectra. Computationally, it is indeed a difficult task to decide if two matrices have a common eigenvalue if that eigenvalue is ill-conditioned. This can be seen from the following discussion.

Let λ and δ be the eigenvalues of A and $A + BK$, respectively. We know that a computed eigenvalue $\tilde{\lambda}$ of A is an eigenvalue of the perturbed matrix $\tilde{A} = A + \Delta A$, where $\|\Delta A\|_2 \leq \mu \|A\|_2$. Similarly, a computed eigenvalue $\tilde{\delta}$ of $A + BK$ is an eigenvalue of $A + BK + \Delta A'$, where $\|\Delta A'\|_2 \leq \mu \|A + BK\|_2$. Thus, even if $\lambda = \delta$, $\tilde{\lambda}$ can be very different from λ and $\tilde{\delta}$ very different from δ , implying that $\tilde{\lambda}$ and $\tilde{\delta}$ are different.

Example 6.6.2 Consider the following example due to Paige (1981) where the matrices A and B are taken as

$$A = Q^T \hat{A} Q, \quad B = Q^T \hat{B},$$

Here \hat{A} is the well-known 20×20 Wilkinson bidiagonal matrix

$$\hat{A} = \begin{pmatrix} 20 & 20 & & 0 & & \\ & 19 & 20 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & 20 & \\ 0 & & & & & 1 \end{pmatrix},$$

$$\hat{B} = (1, 1, \dots, 1, 0)^T,$$

and Q is the Q -matrix of the QR factorization of a randomly generated 20×20 arbitrary matrix whose entries are uniform random numbers on $(-1, 1)$.

Clearly, the pair (\hat{A}, \hat{B}) , and therefore, the pair (A, B) , are uncontrollable. (Note that controllability or uncontrollability is preserved by nonsingular transformations).

Now taking K as a 1×20 matrix with entries as random numbers uniformly distributed on $(-1, 1)$, the eigenvalues $\tilde{\lambda}_i$ of A , and μ_i of $A + BK$ were computed and tabulated. They are displayed in the following table.

In this table, $\rho(B, A - \tilde{\lambda}_i I)$ denotes the ratio of the smallest to the largest singular value of the matrix $(B, A - \tilde{\lambda}_i I)$.

TABLE

Eigenvalues $\tilde{\lambda}_i(A)$	Eigenvalues $\mu_i(A + BK)$	$\rho(B, A - \tilde{\lambda}_i I)$
$-0.32985 \pm j1.06242$	$0.99999 \pm j0$.002
$0.9219 \pm j3.13716$	$-8.95872 \pm j3.73260$.004
$3.00339 \pm j3.13716$	$-5.11682 \pm j9.54329$.007
$5.40114 \pm j6.17864$	$-.75203 \pm j14.148167$.012
$8.43769 \pm j7.24713$	$5.77659 \pm j15.58436$.018
$11.82747 \pm j7.47463$	$11.42828 \pm j14.28694$.026
$15.10917 \pm j6.90721$	$13.30227 \pm j12.90197$.032
$18.06886 \pm j5.66313$	$18.59961 \pm j14.34739$.040
$20.49720 \pm j3.81950$	$23.94877 \pm j11.80677$.052
$22.06287 \pm j1.38948$	$28.45618 \pm j8.45907$.064

The table shows that $\tilde{\lambda}_i$ are almost unrelated to μ_i . One will, then, erroneously conclude that the pair (A, B) is controllable.

The underlying problem, of course, is the ill-conditioning of the eigenvalues of \hat{A} .

Note that, because of ill-conditioning, the computed eigenvalues of A are different from those of \hat{A} , which, in theory, should have been the same; because A and \hat{A} are similar.

The entries of the third column of the table can be used to illustrate the difficulty with the eigenvalue criterion (Criterion (v) of Theorem 6.2.1).

Since the pair (A, B) is uncontrollable, by the eigenvalue criterion of controllability, $\text{rank}(B, A - \tilde{\lambda}_i I)$, for some $\tilde{\lambda}_i$, should be less than n ; consequently, one of the entries of the third column should be identically zero. But this is not the case, **only there is an indication that some “modes” are less controllable than the others.**

To confirm the fact that ill-conditioning of the eigenvalues of \hat{A} is indeed the cause of such failure, $\text{rank}(B, A - I)$, which corresponds to the exact eigenvalue 1 of \hat{A} was computed and seen to be

$$\text{rank}(B, A - I) = 5 \times 10^{-8}.$$

Thus, this test would have done well if the exact eigenvalues of \hat{A} were used, in place of the computed eigenvalues of A , which are complex.

6.7 A Numerically Effective Test of Controllability

A numerically effective test of controllability can be obtained through the reduction of the pair (A, B) to a *block Hessenberg form* using **orthogonal similarity transformation**. The process constructs an orthogonal matrix P such that

$$\begin{aligned} PAP^T &= H, \text{ a block upper Hessenberg matrix} \\ PB &= \tilde{B} = \begin{pmatrix} B_1 \\ 0 \end{pmatrix}. \end{aligned} \tag{6.7.1}$$

The form (6.7.1) is called the **controller-Hessenberg form** of (A, B) , and the pair (H, \tilde{B}) is called the controller-Hessenberg pair of (A, B) (See Rosenbrock (1970)). The reduction to this form can be done using Householder's or Givens' method. We describe the reduction here using Householder's transformations. The algorithmic procedure was suggested by Boley (1981), Van Dooren and Verhaegen (1985), and Paige (1981), etc. See also Rosenbrock (1970). The algorithm is usually known as the **staircase algorithm**.

Let A be $n \times n$ and B be $n \times m$ ($m \leq n$).

Step 0. Triangularize the matrix B using the QR factorization with column pivoting; that is, find an orthogonal matrix P_1 and a permutation matrix E_1 such that

$$P_1BE_1 = \begin{pmatrix} B_1 \\ 0 \end{pmatrix},$$

where B_1 is an $n_1 \times m$ upper triangular matrix and $n_1 = \text{rank}(B) = \text{rank}(B_1)$.

Step 1. Update A and B ; that is, compute

$$P_1AP_1^T = H = \begin{pmatrix} H_{11}^{(1)} & H_{12}^{(1)} \\ H_{21}^{(1)} & H_{22}^{(1)} \end{pmatrix}, \quad \tilde{B} = P_1B = \begin{pmatrix} B_1 \\ 0 \end{pmatrix} E_1^T \equiv \begin{pmatrix} B_1 \\ 0 \end{pmatrix}.$$

where $H_{11}^{(1)}$ is $n_1 \times n_1$, and $H_{21}^{(1)}$ is $(n - n_1) \times n_1$, $n_1 \leq n$. If $H_{21}^{(1)} = 0$, stop.

Step 2. Triangularize $H_{21}^{(1)}$ using the QR factorization with column pivoting; that is, find an orthogonal matrix \hat{P}_2 and a permutation matrix E_2 such that

$$\hat{P}_2H_{21}^{(1)}E_2 = \begin{pmatrix} H_{21}^{(*)} \\ 0 \end{pmatrix},$$

where $H_{21}^{(*)}$ is $n_2 \times n_1$, $n_2 = \text{rank}(H_{21}^{(1)}) = \text{rank}(H_{21}^{(*)})$, and $n_2 \leq n_1$.

If $n_1 + n_2 = n$, stop.

Form

$$P_2 = \text{diag}\left(I_{n_1}, \hat{P}_2\right) = \left(\begin{array}{c|c} I_{n_1} & 0 \\ \hline 0 & \hat{P}_2 \end{array}\right),$$

where I_{n_1} is a matrix consisting of the first n_1 rows and columns of the identity matrix.

Compute

$$H_2 = P_2 H_1 P_2^T = \begin{pmatrix} H_{11}^{(1)} & H_{12}^{(2)} & H_{13}^{(2)} \\ \hline H_{21}^{(2)} & H_{22}^{(2)} & H_{23}^{(2)} \\ \hline 0 & H_{32}^{(2)} & H_{33}^{(2)} \end{pmatrix}$$

where $H_{22}^{(2)}$ is $n_2 \times n_2$ and $H_{32}^{(2)}$ is $(n - n_1 - n_2) \times n_2$. Note that $H_{21}^{(2)} = H_{21}^{(*)} E_2^T$.

Update $P \equiv P_2 P_1$.

If $H_{32}^{(2)} = 0$, stop. (Note that $H_{11}^{(1)}$ does not change).

The matrix \tilde{B} remains unchanged.

Step 3. Triangularize $H_{32}^{(2)}$ to obtain its rank. That is, find \hat{P}_3 and E_3 such that $\hat{P}_3 H_{32}^{(2)} E_3 = \begin{pmatrix} H_{32}^{(3)} \\ 0 \end{pmatrix}$.

Let $n_3 = \text{rank}(H_{32}^{(2)}) = \text{rank}(H_{32}^{(3)})$; $n_3 \leq n_2$.

If $n_1 + n_2 + n_3 = n$, stop. Otherwise, compute P_3, H_3 , and update P as above. (**Note that \tilde{B} remains unchanged**).

The process is continued until for some integer $k \leq n$, the algorithm produces

$$H \equiv \begin{pmatrix} H_{11} & H_{12} & H_{13} & \cdots & H_{1k} \\ H_{21} & H_{22} & H_{23} & \cdots & H_{2k} \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & H_{k,k-1} & H_{kk} \end{pmatrix}, \quad \tilde{B} \equiv \begin{pmatrix} B_1 \\ 0 \end{pmatrix}, \quad (6.7.2)$$

where, either $H_{k,k-1}$ has full rank n_k , signifying that the pair (A, B) is controllable, or $H_{k,k-1}$ is a zero matrix signifying that the pair (A, B) is uncontrollable.

(Note that in above expressions for H and \tilde{B} , the superscripts have been dropped, for convenience. However, H_{21} stands for $H_{21}^{(2)}$, H_{32} stands for $H_{32}^{(3)}$, etc; that is, $H_{k,k-1}$ is established at step k).

It is easy to see that

$$(\tilde{B}, H\tilde{B}, H^2\tilde{B}, \dots, H^{k-1}\tilde{B}) = P(B, AB, \dots, A^{k-1}B)$$

$$= \begin{pmatrix} B_1 & \cdots & \cdots \\ \hline \boxed{H_{21}B_1} & \cdots & \cdots \\ \hline \boxed{H_{32}H_{21}B_1} & & \cdots \\ & \ddots & \\ & & \boxed{H_{k,k-1} \dots H_{21}B_1} \end{pmatrix}$$

implying that the matrix $H_{k,k-1}$ is of full rank if the system is controllable or is a zero matrix if the system is uncontrollable.

Theorem 6.7.1 (Controller-Hessenberg Theorem) (i) Given the pair (A, B) , the orthogonal matrix P constructed by the above procedure is such that $PAP^T = H$ and $PB = \bar{B}$ where H and \bar{B} are given by (6.7.2).

(ii) If pair (A, B) is controllable, then $H_{k,k-1}$ has full rank. If it is uncontrollable, then $H_{k,k-1} = 0$.

Proof: The proof of Theorem 6.7.1 follows from above construction. However, we will prove here part (ii) using (v) of Theorem 6.2.1.

Obviously, $\text{rank } (B, A - \lambda I) = n$ for all λ if and only if $\text{rank } (\bar{B}, H - \lambda I) = n$ for all λ .

Now

$$(\tilde{B}, H - \lambda I) = \begin{pmatrix} B_1 & H_{11} - \lambda I_1 & \cdots & \cdots & H_{1k} \\ 0 & H_{21} & H_{22} - \lambda I_2 & \cdots & H_{2k} \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ 0 & 0 & \cdots & 0 & H_{k,k-1} & H_{kk} - \lambda I_k \end{pmatrix}.$$

If the system is controllable, then the matrix $(\tilde{B}, H - \lambda I)$ must have full rank and thus, the matrix $H_{k,k-1}$ has full rank. On the other hand, if the system is not controllable, then the matrix $(\tilde{B}, H - \lambda I)$ cannot have full rank implying that $H_{k,k-1}$ must be a zero matrix. ■

Notes:

- (i) The matrix \bar{B} is not affected throughout the whole process.
- (ii) At each step of computation, the rank of a matrix has to be determined. We have used QR factorization with column pivoting for this purpose. However, **the best way to do this is to use singular value decomposition of that matrix;** (See Golub and Van Loan (1996), or Datta (1995)).
- (iii) From the construction of the block Hessenberg pair (H, \tilde{B}) , it follows that as soon as we encounter a zero block on the subdiagonal of H or if the matrix B_1 does not have full rank, we stop, concluding that (A, B) is not controllable.

Example 6.7.1 (An uncontrollable pair).

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Step 0. $P_1 = \begin{pmatrix} -0.5774 & -0.5774 & -0.5774 \\ 0.8165 & -0.4082 & -0.4082 \\ 0 & -0.7071 & 0.7071 \end{pmatrix}$, $E_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$,

$$P_1 BE_1 = \begin{pmatrix} -1.7321 & -1.7321 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Step 1. $H_1 = P_1 A P_1^T = \begin{pmatrix} 2.3333 & 0.2357 & -0.4082 \\ -0.4714 & 0.1667 & -0.2887 \\ 0.8165 & -0.2887 & 0.5000 \end{pmatrix}$, $\tilde{B} = \begin{pmatrix} -1.7321 & -1.7321 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$.

Step 2. $H_{21}^{(1)} = \begin{pmatrix} -0.4714 \\ 0.8165 \end{pmatrix}$

$$\hat{P}_2 = \begin{pmatrix} -0.5000 & 0.8660 \\ 0.8660 & 0.5000 \end{pmatrix}, E_2 = 1.$$

$$P_2 = \text{diag}(I_1, \hat{P}_2)$$

$$H_2 = P_2 H_1 P_2^T = \begin{pmatrix} 2.3333 & -0.4714 & 0 \\ 0.9428 & 0.6667 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Since $H_{32}^{(2)} = 0$, we stop.

The controller-Hessenberg form is given by ($H \equiv H_2, \tilde{B}$)

$$H = \begin{pmatrix} 2.3333 & -0.4714 & 0 \\ 0.9428 & 0.6667 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\tilde{B} = \begin{pmatrix} -1.7321 & -1.7321 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Clearly the pair (A, B) is not controllable.

Example 6.7.2 (A controllable pair)

$$A = \begin{pmatrix} 0.7665 & 0.1665 & 0.9047 & 0.4540 & 0.5007 \\ 0.4777 & 0.4865 & 0.5045 & 0.2661 & 0.3841 \\ 0.2378 & 0.8977 & 0.5163 & 0.0907 & 0.2771 \\ 0.2749 & 0.9092 & 0.3190 & 0.9478 & 0.9138 \\ 0.3593 & 0.0606 & 0.9866 & 0.0737 & 0.5297 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.4644 & 0.8278 \\ 0.9410 & 0.1254 \\ 0.0501 & 0.0159 \\ 0.7615 & 0.6885 \\ 0.7702 & 0.8682 \end{pmatrix}.$$

Step 0.

$$P_1 = \begin{pmatrix} -0.3078 & -0.6236 & -0.0332 & -0.5047 & -0.5104 \\ 0.5907 & -0.7058 & -0.0263 & 0.1485 & 0.3610 \\ -0.0080 & -0.0451 & 0.9989 & -0.0047 & -0.0004 \\ -0.4561 & -0.2970 & -0.0132 & 0.8208 & -0.1728 \\ -0.5901 & -0.1510 & -0.0123 & -0.2225 & 0.7611 \end{pmatrix}, E_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$P_1 B E_1 = \begin{pmatrix} -1.5089 & -1.1242 \\ 0 & 0.8157 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$n_1 = \text{rank}(B) = 2$$

Step 1.

$$H_1 = P_1 A P_1^T = \begin{pmatrix} 1.8549 & -0.3935 & -1.2228 & 0.1796 & -0.0198 \\ -0.3467 & 0.3934 & 0.5690 & 0.0593 & 0.0470 \\ -0.7857 & -0.4020 & 0.4421 & -0.3453 & -0.0848 \\ -0.5876 & -0.3573 & -0.4998 & 0.3311 & 0.2607 \\ 0.6325 & -0.0777 & 0.0837 & -0.1295 & 0.2253 \end{pmatrix}$$

$$= \begin{pmatrix} H_{11}^{(1)} & H_{12}^{(1)} \\ H_{21}^{(1)} & H_{22}^{(1)} \end{pmatrix}.$$

$$\tilde{B} = \begin{pmatrix} -1.5089 & -1.1242 \\ 0 & 0.8127 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Step 2.

$$\hat{P}_2 = \begin{pmatrix} -0.6731 & -0.5034 & 0.5418 \\ -0.3545 & -0.4233 & -0.8337 \\ 0.6490 & -0.7533 & 0.1064 \end{pmatrix}, E_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\hat{P}_2 H_{21}^{(1)} = \begin{pmatrix} H_{21}^{(2)} \\ 0 \end{pmatrix} = \begin{pmatrix} 1.1674 & 0.4084 \\ 0 & 0.3585 \\ 0 & 0 \end{pmatrix}$$

$$n_2 = \text{rank} \left(H_{21}^{(1)} \right) = \text{rank} \left(H_{21}^{(2)} \right) = 2.$$

$$P_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -0.6731 & -0.5034 & 0.5418 \\ 0 & 0 & -0.3545 & -0.4233 & -0.8337 \\ 0 & 0 & 0.6490 & -0.7533 & 0.1064 \end{pmatrix}$$

$$P = \begin{pmatrix} -0.3078 & -0.6236 & -0.0332 & -0.5047 & -0.5104 \\ 0.5907 & -0.7058 & -0.0263 & 0.1485 & 0.3610 \\ -0.0847 & 0.0980 & -0.6724 & -0.5306 & 0.4996 \\ 0.6879 & 0.2676 & -0.3383 & -0.1602 & -0.5613 \\ 0.2756 & 0.1784 & 0.6570 & -0.6450 & 0.2109 \end{pmatrix}$$

$$H_2 = P_2 H_1 P_2^T = \begin{pmatrix} 1.8549 & -0.3935 & 0.7219 & 0.3740 & -0.9310 \\ -0.3467 & 0.3934 & -0.3874 & -0.2660 & 0.3297 \\ 1.1674 & 0.4084 & 0.0286 & -0.0378 & 0.0080 \\ 0 & 0.3585 & -0.1807 & 0.1907 & -0.1062 \\ 0 & 0 & -0.3304 & 0.1575 & 0.7792 \end{pmatrix}$$

$$= \begin{pmatrix} H_{11}^{(1)} & H_{12}^{(2)} & H_{13}^{(2)} \\ H_{21}^{(2)} & H_{22}^{(2)} & H_{23}^{(2)} \\ 0 & H_{32}^{(2)} & H_{33}^{(2)} \end{pmatrix}$$

$$\bar{B} = \begin{pmatrix} -1.5089 & -1.1242 \\ 0 & 0.8157 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$n_3 = \text{rank} \left(H_{32}^{(2)} \right) = 1.$$

Since $n_1 + n_2 + n_3 = 2 + 2 + 1 = 5$, we stop.

The controller-Hessenberg form (A, B) is given by $(H \equiv H_2, \bar{B})$.

The pair (A, B) is controllable, because $H_{21}^{(2)}$ and $H_{32}^{(2)}$ have full rank.

Example 6.7.3

$$A = \begin{pmatrix} 0.7665 & 0.1665 & 0.9047 & 0.4540 & 0.5007 \\ 0.4777 & 0.4865 & 0.5045 & 0.2661 & 0.3841 \\ 0.2378 & 0.8977 & 0.5163 & 0.0907 & 0.2771 \\ 0.2749 & 0.9092 & 0.3190 & 0.9478 & 0.9138 \\ 0.3593 & 0.0606 & 0.9866 & 0.0737 & 0.5297 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.4644 & 0.8278 \\ 0.9410 & 0.1254 \\ 0.0501 & 1.0159 \\ 0.7615 & 0.6885 \\ 0.7702 & 0.8682 \end{pmatrix}$$

Step 0.

$$P_1 = \begin{pmatrix} -0.4811 & -0.0729 & -0.5904 & -0.4001 & -0.5046 \\ 0.0213 & -0.7765 & 0.4917 & -0.3183 & -0.2312 \\ -0.6160 & 0.4529 & 0.6231 & -0.0783 & -0.1450 \\ -0.3829 & -0.3429 & -0.0646 & 0.8375 & -0.1739 \\ -0.4919 & -0.2627 & -0.1313 & -0.1764 & 0.8004 \end{pmatrix}$$

$$E_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Step 1.

$$\bar{B} = P_1 B = \begin{pmatrix} -1.0149 & -1.7207 \\ -1.1166 & -0.0000 \\ -0.0000 & 0.0000 \\ 0.0000 & 0.0000 \\ 0.0000 & 0.0000 \end{pmatrix}$$

$$n_1 = 2$$

$$H_1 = P_1 A P_1^T = \begin{pmatrix} 2.1262 & 0.5717 & -0.6203 & 0.3595 & 0.1402 \\ 0.9516 & 0.2348 & -0.0160 & -0.0300 & -0.0472 \\ 0.2778 & -0.4685 & 0.3036 & -0.2619 & -0.0998 \\ 0.0115 & -0.8294 & 0.0243 & 0.3571 & 0.2642 \\ 0.3480 & 0.5377 & 0.0873 & -0.0673 & 0.2250 \end{pmatrix}$$

Step 2.

$$\hat{P}_2 = \begin{pmatrix} -0.4283 & -0.7582 & 0.4916 \\ 0.6685 & 0.1002 & 0.7370 \\ -0.6080 & 0.6442 & 0.4640 \end{pmatrix}$$

$$\begin{aligned}
E_2 &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\
n_2 &= 2 \\
P_2 &= \begin{pmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & -0.4283 & -0.7582 & 0.4916 \\ 0 & 0 & 0.6685 & 0.1002 & 0.7370 \\ 0 & 0 & -0.6080 & 0.6442 & 0.4640 \end{pmatrix} \\
P &= P_2 P_1 \\
P &= \begin{pmatrix} -0.4811 & -0.0729 & -0.5904 & -0.4001 & -0.5046 \\ 0.0213 & -0.7765 & 0.4917 & -0.3183 & -0.2312 \\ 0.3124 & -0.0631 & -0.2824 & -0.6881 & 0.5875 \\ -0.8127 & 0.0749 & 0.3133 & -0.0985 & 0.4755 \\ -0.1004 & -0.6182 & -0.4813 & 0.5053 & 0.3475 \end{pmatrix} \\
H_2 &= P_2 H_1 P_2^T \\
H_2 &= \begin{pmatrix} 2.1262 & 0.5717 & 0.0619 & -0.2753 & 0.6738 \\ 0.9516 & 0.2348 & 0.0064 & -0.0485 & -0.0315 \\ 0.0434 & 1.0938 & 0.1675 & -0.1244 & -0.0811 \\ 0.4433 & 0.0000 & 0.0895 & 0.2539 & -0.2275 \\ -0.0000 & -0.0000 & -0.0517 & 0.1971 & 0.4644 \end{pmatrix} \\
\bar{B} &= \begin{pmatrix} -1.0149 & -1.7207 \\ -1.1166 & -0.0000 \\ -0.0000 & 0.0000 \\ 0.0000 & 0.0000 \\ 0.0000 & 0.0000 \end{pmatrix}
\end{aligned}$$

Flop-Count: Testing controllability using the constructive proof of Theorem 6.7.1 requires roughly $6n^3 + 2n^2m$ flops. The count includes the construction of the transforming matrix P (see Van Dooren and Verhaegen (1985)).

Round-off Error Analysis and Stability: The procedure is **numerically stable**. It can be shown that the computed matrices \hat{H} and \hat{B} are such that $\hat{H} = H + \Delta H$, and $\hat{B} = \bar{B} + \Delta B$, where $\|\Delta H\|_F \leq c\mu\|H\|_F$ and $\|\Delta B\|_F \leq c\mu\|\bar{B}\|_F$ for some small constant c . Thus, with the computed pair (\hat{H}, \hat{B}) , we will compute the controllability of a system determined by the pair of matrices which are close to H and \bar{B} . Since the controllability of the pair (H, \bar{B}) is the same as that of the pair (A, B) , **this can be considered as a backward stable method for finding the controllability of the pair (A, B)** .

MATCONTROL Note: The above method has been implemented in MATCONTROL function **cntrlhs**.

Controllability Index and Controller-Hessenberg Form

Let $B = (b_1, b_2, \dots, b_m)$. Then the controllability matrix C_M can be written as

$$C_M = (b_1, b_2, \dots, b_m; Ab_1, Ab_2, \dots, Ab_m; \dots; A^{n-1}b_1, A^{n-1}b_2, \dots, A^{n-1}b_m).$$

Suppose that the linearly independent columns of the matrix C_M have been obtained in order from left to right. Reorder these independent columns to obtain:

$$C'_M = (b_1, Ab_1, \dots, A^{\mu_1-1}b_1; b_2, Ab_2, \dots, A^{\mu_2-1}b_2; \dots; b_m, Ab_m, \dots, A^{\mu_m-1}b_m).$$

The integers μ_1, \dots, μ_m are called the **controllability indices** associated with b_1, b_2, \dots, b_m , respectively. Note that μ_i is the number of independent columns associated with b_i .

Furthermore, $\mu = \max(\mu_1, \dots, \mu_m)$ is called the **controllability index**. If $\mu_1 + \mu_2 + \dots + \mu_m = n$, then the system is controllable.

It is clear that determining the controllability index is delicate problem from numerical view point; because it is basically a rank-determination problem.

Fortunately, the block-Hessenberg pair (H, \bar{B}) of (A, B) not only determines if the pair (A, B) is controllable but also it gives us the controllability index. In the block-Hessenberg pair (H, \bar{B}) in (6.7.2), k is the controllability index. Thus, for Example 6.7.2, the controllability index is 3.

Controllability Test in the Single-Input Case

In the single-input case, the block Hessenberg form of (A, b) reduces to:

$$PAP^T = H = \begin{pmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2n} \\ 0 & h_{32} & \ddots & \cdots & h_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n,n-1} & h_{nn} \end{pmatrix}, \quad Pb = \bar{b} = \begin{pmatrix} b_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \quad (6.7.3)$$

Theorem 6.7.2 (A, b) is controllable if the controller-Hessenberg pair (H, \bar{b}) is such that H is an unreduced upper Hessenberg, that is, $h_{i,i-1} \neq 0$, $i = 2, \dots, n$, and $b_1 \neq 0$; otherwise, it is uncontrollable.

We will give an independent proof of this test using the Kalman criterion of controllability.

Proof: Observe that

$$\text{rank } (b, Ab, \dots, A^{n-1}b) = \text{rank } (Pb, PAP^T Pb, \dots, PA^{n-1}P^T Pb) = \text{rank } (\bar{b}, H\bar{b}, \dots, H^{n-1}\bar{b}).$$

The last matrix is a lower triangular matrix with $b_1, h_{21}b_1, h_{21}h_{32}b_1, \dots, h_{21} \cdots h_{n,n-1}b_1$ as the diagonal entries.

Since $h_{i,i-1} \neq 0$, $i = 2, \dots, n$ and $b_1 \neq 0$, it follows that $\text{rank } (b, Ab, \dots, A^{n-1}b) = n$.

On the other hand, if any of $h_{i,i-1}$ or b_1 is zero, the matrix $(b, Ab, \dots, A^{n-1}b)$ is rank deficient, and therefore, the system is uncontrollable.

Example 6.7.4 (Example 6.6.2 Revisited): Superiority of the Algorithm over the Other Theoretical Criteria

To demonstrate the superiority of the test of controllability given by Theorem 6.7.2; over some of the theoretical criteria that we considered in the last section, Paige applied the controller-Hessenberg test to the same ill-conditioned problem as in Example 6.6.2. The computations gave $b_1 = 4.35887$, $h_{21} = 8.299699$, $17 < \|h_{i,i-1}\| < 22$, $i = 3, 4, \dots, 9$, and $h_{20,19} = 0.0000027$. Since $h_{20,19}$ is computationally zero in a single-precision computation, the system is uncontrollable, according to the test based on Theorem 6.7.2.

6.8 A Numerically Effective Test of Observability

Analogous to the procedure of obtaining the form (H, \bar{B}) from (A, B) , the pair (A, C) can be transformed to (H, \bar{C}) , where

$$H = OAO^T = \begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1k} \\ H_{21} & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ 0 & & H_{k,k-1} & H_{kk} \end{pmatrix}, \quad (6.8.1)$$

$$\bar{C} = CO^T = (0, C_1).$$

The pair (A, C) is observable if H is block unreduced (that is, all the subdiagonal blocks have full rank) and the matrix C_1 has full rank.

The pair (H, \bar{C}) is said to be an **observer-Hessenberg pair**.

Flop-Count. The construction of the observer-Hessenberg form this way requires roughly $6n^3 + 2n^2r$ flops.

Single-output case

In the single-output case, i.e., when C is a row vector, the pair (A, C) is observable if H is an upper Hessenberg matrix and $\bar{C} = (0, \dots, 0, c_1)$; $c_1 \neq 0$

MATCONTROL Note: MATCONTROL function **obserhs** can be used to obtain the reduction (6.8.1).

6.9 Distance to an Uncontrollable System

The concepts of controllability and observability are generic ones. Since determining if a system is controllable depends upon whether or not a certain matrix (or matrices) has full rank, it is immediately obvious from our discussion on numerical rank of a matrix in Chapter 4 that

any uncontrollable system is arbitrary close to a controllable system. To illustrate this, let us consider the following well-known example (Eising (1984)):

$$A = \begin{pmatrix} -1 & -1 & \cdot & \cdot & \cdot & \cdot & -1 & -1 \\ 1 & \cdot & & & & & -1 & \\ & 1 & \cdot & & & & -1 & \\ & & \cdot & \cdot & & & \cdot & \\ & & & \cdot & \cdot & & \cdot & \\ & & & & \cdot & & \cdot & \\ 0 & & & & \cdot & & -1 & \\ & & & & & 1 & 1 & \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{pmatrix}. \quad (6.9.1)$$

The pair (A, b) is obviously controllable. However, it is easily verified that if we add $(-2^{1-n}, -2^{1-n}, \dots, -2^{1-n})$ to the last row of (B, A) , we obtain an uncontrollable system. Clearly, when n is large, the perturbation 2^{1-n} is small, implying that the original controllable system (A, B) is close to an uncontrollable system. **Thus, what is important in practice is knowledge of how close a controllable system is to an uncontrollable one rather than determining if a system is controllable or not.** To this end, we introduce, following Paige (1981), **a measure of the distance to uncontrollability**, denoted by $\mu(A, B)$:

$$\mu(A, B) \equiv \min \{\|\Delta A, \Delta B\|_2 \text{ such that the system defined by } (A + \Delta A, B + \Delta B) \text{ is uncontrollable}\}.$$

Here ΔA and ΔB are allowable perturbations over a field F . If the field F is \mathbb{R} , then we will use the symbol $\mu_R(A, B)$ to distinguish it from the $\mu(A, B)$.

The quantity $\mu(A, B)$ gives us a measure of the distance of a controllable pair (A, B) to the nearest uncontrollable pair. **If this distance is small, then the original controllable system is close to an uncontrollable system. If this distance is large, then the system is far from an uncontrollable system.**

Here is a well-known result on $\mu(A, B)$. The result was first obtained by Miminis (1981). See also Eising (1984) and Kenney and Laub (1988). **Unless otherwise stated, the perturbations are assumed to be over the field of complex numbers; that is, $F = \mathbb{C}$.**

Theorem 6.9.1 (Singular Value Characterization to Distance to Uncontrollability)
 $\mu(A, B) = \min \sigma_n(sI - A, B)$, where $\sigma_n(sI - A, B)$ is the smallest singular value of $(sI - A, B)$ and s runs over all complex numbers.

Proof: Suppose that $(A + \Delta A, B + \Delta B)$ is an uncontrollable pair. Then according to (v) of Theorem 6.2.1, we have

$$\text{rank}(A + \Delta A - \lambda I, B + \Delta B) < n, \text{ for some } \lambda \in \mathbb{C}.$$

Since the smallest perturbation that can make $\text{rank}(A - \lambda I, B)$ less than n is $\sigma_n(A - \lambda I, B)$, (See section 3.9.3 of Chapter 3), we have

$$\sigma_n(A - \lambda I, B) \leq \|\Delta A, \Delta B\|_2$$

and the equality holds if

$$(\Delta A, \Delta B) = -\sigma_n u_n v_n^*,$$

where σ_n is the smallest singular value of $(A - \lambda I, B)$, and u_n and v_n are the corresponding left and right singular vectors. Taking now the minimum over all $\lambda \in \mathbb{C}$, and using criterion (v) of Theorem 6.2.1, we obtain the result.

Algorithms for Computing $\mu(A, B)$

Based on Theorem 6.9.1, several algorithms (Miminis (1981), Eising (1984), Boley (1987), and Wicks and De Carlo (1991)), have been developed in the last few years to compute $\mu(A, B)$ and $\mu_R(A, B)$.

We will briefly describe here a Newton algorithm due to Elsner and He (1991), and an algorithm due to Wick and DeCarlo (1991).

6.9.1 Newton's and the Bisection Methods for Computing the Distance to Uncontrollability

Let's denote $\sigma_n[sI - A, B]$ by $\sigma(s)$. The problem of finding $\mu(A, B)$ is then clearly the problem of minimizing $\sigma(s)$ over the complex plane.

To this end, define

$$f(s) = v_n^*(s) \begin{pmatrix} u_n(s) \\ 0 \end{pmatrix},$$

where $u_n(s)$ and $v_n(s)$ are the normalized n th columns of U and V in the singular value decomposition of $(A - sI, B)$; that is, $(A - sI, B) = U\Sigma V^T$. The function $f(s)$ plays an important role. The first and second derivatives of $\sigma(s) = \sigma(x + jy) = \sigma(x, y)$ can be calculated using this singular value decomposition. It can be shown that, if $s = x + jy$, then

$$\frac{\partial \sigma}{\partial x} = \frac{\partial \sigma(x + jy)}{\partial x} = -\operatorname{Re} f(x + jy),$$

and

$$\frac{\partial \sigma}{\partial y} = \frac{\partial \sigma(x + jy)}{\partial y} = -\operatorname{Im} f(x + jy).$$

Knowing the first derivatives, the second derivatives can easily be calculated.

Hence the zeros of $f(s)$ are the critical points of the function $\sigma(s)$.

Thus, some well-established root-finding methods, such as **Newton's method**, or the **Bisection method** can be used to compute these critical points.

An interesting observation about the critical points is:

The critical points satisfy $s = u_n^*(s)Au_n(s)$, and hence they lie in the field of values of A .

The result follows from the fact that $\sigma(s)f(s) = u_n^*(s)(A - sI)u_n(s)$, since $(A - sI, B)^*u_n(s) = \sigma(s)v_n(s)$. (For the definition of field of values, see Horn and Johnson (1985)).

To decide which critical points are local minima, one can use the following well-known criterion. A critical point $s = x_c + jy_c$ of $\sigma(s)$ is a local minimum of $\sigma(x, y)$ if

$$\left(\frac{\partial^2 \sigma}{\partial x^2} \right) \left(\frac{\partial^2 \sigma}{\partial y^2} \right) - \left(\frac{\partial^2 \sigma}{\partial x \partial y} \right)^2 > 0 \quad \text{and} \quad \frac{\partial^2 \sigma}{\partial x^2} > 0.$$

Another sufficient condition is:

If $\sigma_{n-1}(A - sI, B) > \sqrt{5}\sigma_n(A - sI, B)$, where $s = x_c + jy_c$ is a critical point, then (x_c, y_c) is a local minimum point of $\sigma(x, y)$.

Newton's method needs a starting approximation. The local minima of $\sigma(x, y)$, generally, are simple. Since all critical points s satisfy: $u_n^*Au_n = s$, all minimum points $s = x + jy$ will lie in the field of values of A ; and hence

$$\begin{aligned} \lambda_{\min} \left(\frac{A + A^T}{2} \right) \leq x \leq \lambda_{\max} \left(\frac{A + A^T}{2} \right) \\ \text{and} \\ \lambda_{\min} \left(\frac{A - A^T}{2j} \right) \leq y \leq \lambda_{\max} \left(\frac{A - A^T}{2j} \right), \end{aligned}$$

where $\lambda_{\max}(C)$ and $\lambda_{\min}(C)$ denote the largest and smallest eigenvalues of the matrix C . Furthermore, since $\sigma_n(A - sI, B) = \sigma_n(A - \bar{s}I, B)$, the search for all local minimum points can be restricted to $0 \leq y \leq \lambda_{\max} \left(\frac{A - A^T}{2j} \right)$.

Based on the above discussion, we now state Newton's algorithms for finding $\mu(A, B)$. Denote $x_k + jy_k$ by $\begin{pmatrix} x_k \\ y_k \end{pmatrix}$.

Algorithm 6.9.1 Newton's Algorithm For Distance to Uncontrollability

Inputs: The matrices A and B .

Output: A local minimum of $\sigma(s)$.

Step 0. Choose $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ using the above criterion.

Step 1. For $k = 0, 1, 2, \dots$ do until convergence

$$\begin{aligned} \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} &= \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \theta_k \begin{pmatrix} p_{k_1} \\ p_{k_2} \end{pmatrix}, \\ \text{where } \begin{pmatrix} p_{k_1} \\ p_{k_2} \end{pmatrix} &= \begin{pmatrix} \operatorname{Re} \frac{\partial f}{\partial x} & \operatorname{Re} \frac{\partial f}{\partial y} \\ \operatorname{Im} \frac{\partial f}{\partial x} & \operatorname{Im} \frac{\partial f}{\partial y} \end{pmatrix}^{-1} \begin{pmatrix} \operatorname{Re} f(x, y) \\ \operatorname{Im} f(x, y) \end{pmatrix}, \end{aligned}$$

choosing θ_k such that $\sigma(x_k - \theta_k p_{k1}, y_k - \theta_k p_{k2}) = \min_{-1 \leq \theta \leq 1} \sigma(x_k - \theta p_{k1}, y_k - \theta p_{k2})$.

End.

Step 2. If $s_c = \begin{pmatrix} x_f \\ y_f \end{pmatrix}$ is the final point upon conclusion of Step 2, then compute the smallest singular value σ_n of the matrix $(A - s_c I, B)$, and take σ_n as the local minimum of $\sigma(s)$.

Choosing θ_k

θ_k can be chosen using again Newton's algorithm, as follows:

Define $g(\theta) = p_{k1} R e f(\theta) + p_{k2} I m f(\theta)$. Then $g'(\theta) = p_{k1} R e f'(\theta) + p_{k2} I m f'(\theta)$.

Newton's Algorithm for Computing θ_k

Step 1. Choose $\theta_0 = 1$

Step 2. For $j = 1, 2, \dots$ do until convergence

$$\theta_{j+1} = \theta_j - \eta_j \frac{g(\theta_j)}{g'(\theta_j)},$$

where η_j is chosen such that

$$\sigma(x_k - \theta_{j+1} p_{k1}, y_k - \theta_{j+1} p_{k2}) < \sigma(x_k - \theta_j p_{k1}, y_k - \theta_j p_{k2})$$

End.

Remarks: Numerical experiments suggest that it is necessary to compute θ'_k 's only a few times to get a good initial point and then as soon as it becomes close 1, it can be set to 1. **Newton's method with $\theta_k = 1$ converges quadratically.**

Example 6.9.1 (Elsner and He (1991)). Let

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0.1 & 3 & 5 \\ 0 & -1 & -1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0.1 \\ 0 \end{pmatrix}.$$

$$\lambda_{\max}\left(\frac{A + A^T}{2}\right) = 3.9925, \quad \lambda_{\min}\left(\frac{A + A^T}{2}\right) = -1.85133$$

$$\lambda_{\max}\left(\frac{A - A^T}{2j}\right) = 3.0745, \quad \lambda_{\min}\left(\frac{A - A^T}{2j}\right) = -3.0745$$

Thus all zero points lie in the rectangular region given by $-1.8513 \leq x \leq 3.9925, -3.0745 \leq y \leq 3.0745$. Choose $x_0 = 1.5$ and $y_0 = 1$.

Then $s_0 = 1.5 + j$.

$\theta_0 = 0.09935, \theta_1 = 0.5641, \theta_2 = 1.0012$. Starting from here, θ_k was set to 1.

The method converged in 5 steps. $s_c = \begin{pmatrix} x_5 \\ y_5 \end{pmatrix} = \begin{pmatrix} 0.93708 \\ 0.998571 \end{pmatrix} = 0.93708 + 0.998571j$.

The minimum singular value of $(A - s_c I, B) = 0.0392$.

Thus $\mu(A, B) = 0.039238$

Computing $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$

$\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ can be computed using the following formulae:

$$\begin{aligned}\frac{\partial f}{\partial x} &= v_{dx}^* \begin{pmatrix} u_n \\ 0 \end{pmatrix} + v_n^* \begin{pmatrix} u_{dx} \\ 0 \end{pmatrix} + \frac{j}{\sigma_n} (\text{Im } f) f, \\ \frac{\partial f}{\partial y} &= v_{dy}^* \begin{pmatrix} u_n \\ 0 \end{pmatrix} + v_n^* \begin{pmatrix} u_{dy} \\ 0 \end{pmatrix} - \frac{j}{\sigma_n} (\text{Re } f) f,\end{aligned}$$

where u_{dx} and v_{dx} are given by

$$\begin{aligned}u_{dx} &= \sum_{k=1}^{n-1} \alpha_{xk} u_k, \\ v_{dx} &= \sum_{k=1}^{n-1} \beta_{xk} v_k - \frac{1}{\sigma_n} \sum_{k=n+1}^{n+m} \left(v_k^* \begin{pmatrix} u_n \\ 0 \end{pmatrix} \right) v_k,\end{aligned}$$

where

$$\alpha_{xk} = \frac{\sigma_n [u_k^*, 0] v_n + \sigma_k v_k^* \begin{pmatrix} u_n \\ 0 \end{pmatrix}}{\sigma_k^2 - \sigma_n^2}, \quad \beta_{xk} = \frac{\sigma_k [u_k^*, 0] v_n + \sigma_n v_k^* \begin{pmatrix} u_n \\ 0 \end{pmatrix}}{\sigma_k^2 - \sigma_n^2}.$$

Similarly,

$$\begin{aligned}u_{dy} &= j \sum_{k=1}^{n-1} \alpha_{yk} u_k, \\ v_{dy} &= j \sum_{k=1}^{n-1} \beta_{yk} v_k + \frac{j}{\sigma_n} \sum_{k=n+1}^{n+m} \left(v_k^* \begin{pmatrix} u_n \\ 0 \end{pmatrix} \right) v_k,\end{aligned}$$

where

$$\alpha_{yk} = \frac{\sigma_n [u_k^*, 0] v_n - \sigma_k v_k^* \begin{pmatrix} u_n \\ 0 \end{pmatrix}}{\sigma_k^2 - \sigma_n^2}, \quad \beta_{yk} = \frac{\sigma_k [u_k^*, 0] v_n - \sigma_n v_k^* \begin{pmatrix} u_n \\ 0 \end{pmatrix}}{\sigma_k^2 - \sigma_n^2}.$$

MATLAB Note. MATLAB codes for Algorithm 6.9.1 are available from the authors of the paper.

The Bisection Method (Real Case)

In the real case, the following bisection method can also be used to compute the zeros of $f(s)$.

Step 1. Find an interval $[a, b]$ such that $f(a)f(b) < 0$.

Step 2.

Step 2.1 Compute $c = \frac{a+b}{2}$.

Step 2.2 If $f(c)f(b) < 0$, then set $a = c$ and return to Step 2.1.

If $f(a)f(c) < 0$, then set $b = c$ and return to Step 2.1.

Step 3. Repeat Step 2 until c is an acceptable zero point of $f(s)$.

6.9.2 The Wicks-DeCarlo Method for Distance to Uncontrollability

Newton's algorithm, described in Section 6.9.1 is based on minimization of $\sigma_n(sI - A, B)$ over all complex numbers s . It requires an SVD computation at each iteration.

In this section, we state an algorithm due to Wicks and DeCarlo (1991). The algorithm is also iterative in nature but “requires only two QR factorizations at each iteration without the need for searching or using a general minimization algorithm.”

The algorithm is based on the following observation:

$$\mu(A, B) = \min_{u \in \mathbb{C}^n} \|(u^* A(I - uu^*) u^* B\|, \quad (6.9.2)$$

subject to $u^* u = 1$. Based on this observation, they developed three algorithms for computing $\mu_R(A, B)$ and $\mu(A, B)$.

We state here why one of the algorithms (Algorithm II in Wicks and DeCarlo (1991)) for computing $\mu(A, B)$.

Definition 6.9.1 Let the distance measure $d_1(A, B)$ be defined by

$$\begin{aligned} [d_1(A, B)]^2 &= \|[e_n^*(A(I - e_n e_n^*) B)]\|_2^2 \\ &= \sum_{j=1}^{n-1} |a_{nj}|^2 + \sum_{j=1}^m |b_{nj}|^2. \end{aligned}$$

Using the above notation, it has been shown in Wicks and DeCarlo (1991) that

$$\mu(A, B) = \min_{\substack{U \in \mathbb{C}^{n \times n} \\ U^* U = I}} d_1(U^* A U, U^* B)$$

The algorithm proposed by them constructs a sequence of unitary matrices U_1, U_2, \dots , such that

1. $A_{k+1} = U_k^* A_k U_k, B_{k+1} = U_k^* B$
2. $d_1(A_{k+1}, B_{k+1}) < d_1(A_k, B_k)$
3. $\lim_{k \rightarrow \infty} d_1(A_k, B_k)$ is a local minimum of (6.9.2).

Algorithm 6.9.2 An Algorithm for Computing $\mu(A, B)$

Inputs: The matrices A and B

Output: $\mu(A, B)$.

Step 0. Ste $A_1 \equiv A, B_1 \equiv B$.

Step 1. For $k = 1, 2, \dots$ until convergence.

Step 1.1. Form $M_k = (A_k - (a_{nn})_k I \ B_k)$.

Step 1.2. Factor $M_k = L_k V_k$, where L_k is lower triangular and V_k is unitary.

Step 1.3. Find the QR factorization of $L_k = U_k^* R_k$.

Step 1.4. Set $A_{k+1} = U_k^* A_k U_k$, $B_{k+1} = U_k^* B_k$.

Step 1.5. If $d_1(A_{k+1}, B_{k+1}) = d_1(A_k, B_k)$, stop.

End.

Proof: The proof amounts to showing that $d_1(A_k, B_k) \geq |r_{nn}|$, where $R_k = \begin{pmatrix} r_{11} & \dots & \dots & r_{1n} \\ 0 & r_{22} & & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & & & r_{nn} \end{pmatrix}$

and as such $|r_{nn}| = d_1(U_k^* A_k U_k, U_k^* B_k)$.

For details, the readers are referred to the paper of Wicks and DeCarlo (1991).

Example 6.9.2

$$A = \begin{pmatrix} 0.950 & 0.891 & 0.821 & 0.922 \\ 0.231 & 0.762 & 0.445 & 0.738 \\ 0.607 & 0.456 & 0.615 & 0.176 \\ 0.486 & 0.019 & 0.792 & 0.406 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.9350 & 0.0580 & 0.1390 \\ 0.9170 & 0.3530 & 0.2030 \\ 0.4100 & 0.8130 & 0.1990 \\ 0.8940 & 0.0100 & 0.6040 \end{pmatrix}. \quad \text{Let the tolerance for stopping the iteration be: Tol} = 0.00001.$$

Define $\mu_k = d_1(A_k, B_k)$.

The algorithm produces the following converging sequence of μ_k :

k	μ_k	k	μ_k
0	1.42406916966838	10	0.41450782001833
1	0.80536738314449	11	0.41450781529413
2	0.74734006994998	12	0.41450781480559
3	0.52693889988172	13	0.41450781475487
4	0.42241562062172	14	0.41450781474959
5	0.41511102322896	15	0.41450781474904
6	0.41456112538077	16	0.41450781474899
7	0.41451290008455	17	0.41450781474898
8	0.41450831981698	18	0.41450781474898
9	0.41450786602577	19	0.41450781474898

and after 19 iteration the algorithm returns $\mu = 0.41450781474898$.

MATCONTROL Note. Algorithm 6.9.2 has been implemented in MATCONTROL function `discntrl`.

6.9.3 A Global Minimum Search Algorithm.

The algorithms by Elsner and He (1991) and Wicks and DeCarlo (1992) are guaranteed only to converge to a local minimum rather than a global minimum. A global minimum search algorithm was given by Gao and Neumann (1993). Their algorithm is based on the observation that if $\text{rank}(B) < n$, then the minimization problem can be transformed to a minimization problem in the bounded region $\{(x, z) | x \leq \|A\|_2, |z| \leq \|A\|_2\}$ in the two dimensional real plane. The algorithm then progressively partitions this region into simplexes and finds lower and upper bounds for $\mu(A, B)$ by determining if the vertices (x_k, z_k) satisfy

$$z_k > \min_{y \in \mathbb{R}} \sigma_{\min}(A - (x_k + jy)I, B).$$

These bounds are close to each other if $\mu(A, B)$ is small. “If $\mu(A, B)$ is not small, then the algorithm produces a lower bound which is not small, thus leading us to a safe conclusion that (A, B) is not controllable”.

For details of the algorithm, see Gao and Neuman (1993). See also **Example 26**.

6.10 Distance to Uncontrollability and the Singular Values of the Controllability Matrix

Since the rank of the controllability matrix C_M determines whether a system is controllable or not, and the most numerically effective way to determine the rank of a matrix is via the singular values of the matrix, it is natural to wonder, what roles do the singular values of the controllability matrix play in deciding if a given controllable system is near an uncontrollable system. (Note that Theorem 6.9.1 and the associated algorithm for computing $\mu(A, B)$ use the singular values of $(A - sI, B)$).

The following result due to Boley and Lu (1986) sheds some light in that direction. We state the result without proof. Proof can be found in Boley and Lu (1986).

Theorem 6.10.1 *Let (A, B) be a controllable pair. Then*

$$\mu(A, B) \leq \mu_R(A, B) \left(1 + \frac{\|C_p\|}{\sigma_{n-1}} \right) \sigma_n,$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{n-1} \geq \sigma_n$ are the singular values of the controllability matrix $C_M = (B, AB, \dots, A^{n-1}B)$ and C_p is a companion matrix similar to A .

Example 6.10.1 *We consider Example 6.9.1 again.*

The singular values of the controllability matrix are 2.2221, 0.3971, 0.0227.

The companion matrix C_p is calculated as follows:

$$\begin{aligned} x_1 &= (1, 0, 0)^T \\ x_2 &= Ax_1 = (1, 0.1, 0)^T \\ x_3 &= Ax_2 = (1.1, 0.4, -0.1)^T. \end{aligned}$$

Then the matrix $X = (x_1, x_2, x_3)$ is such that

$$X^{-1}AX = C_p = \begin{pmatrix} 0 & 0 & 2 \\ 1 & 0 & -3.9 \\ 0 & 1 & 3 \end{pmatrix}$$

According to Theorem 6.10.1, we then have

$$\mu(A, B) \leq \left(1 + \frac{5.3919}{0.3971}\right) \times 0.02227 = 0.3309.$$

Remarks: The above theorem can be used to predict the order of perturbations needed to transform a controllable system to an uncontrollable system. *It is the largest gap between the consecutive singular values.* (However, note that, in general, the singular values of the controllability matrix cannot be used directly to make a prediction of how close the system is to an uncontrollable system).

In other words, it is **not** true that one can obtain a nearly uncontrollable system by applying perturbations $\Delta A, \Delta B$, with norm bounded by the smallest nonzero singular value of the controllability matrix.

6.11 Some Selected Software

6.11.1 MATLAB CONTROL SYSTEM TOOLBOX

canon—State-space canonical forms.

ctrb, obsv –Controllability and observability matrices.

gram—Controllability and observability gramians.

ctrbf—Controllability staircase form.

6.11.2 MATCONTROL

CNTRLHS—Finding the controller-Hessenberg form

OBSERHS—Finding the observer-Hessenberg form

CNTRLC—Find the controller canonical form (Lower Companion)

DISCNTRL—Distance to controllability using the Wicks-DeCarlo algorithm

6.11.3 CSP-ANM

- Reduction to controller-Hessenberg and observer-Hessenberg forms

- Block controller-Hessenberg forms are computed by `controllerHessenbergForm[system]` and `LowercontrollerHessenbergForm [system]`.
- Block observer-Hessenberg forms are computed by `ObserverHessenbergForm [system]` and `UpperObserverHessenbergForm [system]`.

- **Controllability and observability tests**

- Tests of controllability and observability using block controller-Hessenberg and block observer-Hessenberg forms are performed via `Controllable [system, ControllabilityTest → FullRankcontrollerHessenbergBlocks]` and `Observable [system, ObservabilityTest → FullRankObserverHessenbergBlocks]`.
- Tests of controllability and observability of a stable system via positive definiteness of Gramians are performed via `Controllable [system, ControllabilityTest → PositiveDiagonalCholeskyFactorControllabilityGramian]` and `Observable [system, ObservabilityTest → PositiveDiagonalCholeskyFactorObservabilityGramian]`.

6.11.4 SLICOT

Canonical and Quasi Canonical Forms

AB01MD	Orthogonal controllability form for single-input system
AB01ND	Orthogonal controllability staircase form for multi-input system
AB01OD	Staircase form for multi-input system using orthogonal transformations
TB01MD	Upper/lower controller Hessenberg form
TB01ND	Upper/lower observer Hessenberg form
TB01PD	Minimal, controllable or observable block Hessenberg realization
TB01UD	Controllable block Hessenberg realization for a state-space representation
TB01ZD	Controllable realization for single-input systems

6.11.5 MATRIX_X

Purpose: Obtain controllable part of a dynamic system.

Syntax: `[SC, NSC, T]=CNTRLABLE (S, NS, TOL)`

Purpose: Compute observable part of a system.

Syntax: `[SOBS, NSOBS, T]=OBSERVABLE (S, NS, TOL)`

Purpose: Staircase form of a system matrix.

Syntax: `[SST, T, NCO]=STAIR (S, NS, TOL)`

6.12 Summary and Review

I. Algebraic Criteria of Controllability and observability.

Controllability and observability are two most fundamental concepts in control theory. The algebraic criteria of controllability and observability are summarized in **Theorem 6.2.1** and **Theorem 6.3.1**, respectively.

Unfortunately, these algebraic criteria very often do not yield numerically viable tests for controllability and observability. The numerical difficulties with these criteria as practical tests of controllability are discussed and illustrated in Section 6.6. The pair (A, B) in Example 6.6.1 is a controllable pair; however, it is shown that the Kalman criterion of controllability (Criterion (ii) of Theorem 6.2.1) leads to an erroneous conclusion due to a computationally small singular value of the controllability matrix. Similarly, in Example 6.6.2, it is shown how an obviously uncontrollable pair can be taken as a controllable pair by using the eigenvalue criterion of controllability (Criterion (v) of Theorem 6.2.1) as a numerical test, due to the ill-conditioning of the eigenvalues of the matrix A .

II. Numerically effective tests of controllability and observability. Computationally viable tests of controllability and observability are given in **Section 6.7** and **Section 6.8**. These tests are based on the reductions of the pairs (A, B) and (A, C) , respectively, to **controller-Hessenberg** and **observer-Hessenberg** forms. These forms can be obtained by using orthogonal transformations and the tests can be shown to be numerically stable.

Indeed, when controller-Hessenberg test is applied to Example 6.6.2, it was concluded correctly that, in spite of the ill-conditioning of the eigenvalues of A , the pair (A, B) is uncontrollable.

III. Distance to Uncontrollability. Since determining the rank of a matrix is numerically a delicate problem and the problem is sensitive to small perturbations, **in practice, it is more important to find when a controllable system is close to an uncontrollable system.** To this end, a practical measure of the distance to uncontrollability, denoted by $\mu(A, B)$, is introduced in Section 6.9: $\mu(A, B) = \min\{\|\Delta A, \Delta B\|_2\}$ such that the pair $(A + \Delta A, B + \Delta B)$ is controllable.

A well-known characterization of $\mu(A, B)$ is given in **Theorem 6.9.1**. This theorem states:

$$\mu(A, B) = \min \sigma_n(sI - A, B), \text{ where } \sigma_n(sI - A, B) \text{ is the smallest singular value of the matrix } (sI - A, B) \text{ and } s \text{ runs over all complex numbers.}$$

Two algorithms (**Algorithm 6.9.1** and **Algorithm 6.9.2**), have been described to compute $\mu(A, B)$.

6.13 Chapter Notes and Further Reading

Controllability and observability are two most basic concepts in control theory. The results related to controllability and observability can be found in any standard book in linear systems (e.g., Brockett (1970), Chen (1984), Kailath (1980), Brogan (1991), De Carlo (1989), Kalman, et. al (1969), Rosenbrock (1970), Luenberger (1979), etc.).

For details on the staircase algorithms for finding the controller-Hessenberg and observer-Hessenberg forms, see Boley (1981), Paige (1981), Van Dooren and Verhaegen (1985). For computation of the Kalman decomposition, see Boley (1980), Boley (1991), etc. For more on the concept of the distance to uncontrollability and related algorithms, see Boley (1987), Boley and Lu (1986), Eising (1984), Wicks and De Carlo (1991), Elsner and He (1991), Paige (1981), Miminis (1981), Kenney and Laub (1988), and Gao and Neumann (1993). For a test of controllability via real Schur form, see Varga (1979).

EXERCISES

1. Prove that (A, B) is controllable if and only if for a constant matrix F , the matrix $(A + BF, B)$ is controllable; that is, the controllability of a system does not change under state feedback. (**The concept of state feedback is defined in Chapter 10.**)
2. Construct an example to show that the observability of a system may change under state feedback.
3. A matrix A is called a **cyclic matrix** if in the Jordan canonical form of A , there is only one Jordan box associated with each distinct eigenvalue.

Let A be a cyclic matrix and let the pair (A, B) be controllable. Then prove that for almost all vectors v , the pair (A, Bv) is controllable.

4. Give an 2×2 example to show that the cyclicity assumption is essential for the result of Problem 3 to hold.

5. Show that (A, c) is observable if and only if there exists a vector k such that $\left(\begin{pmatrix} c \\ k \end{pmatrix}, A - bk \right)$ is observable.

6. Prove the Parts (i), (ii), (iv), (v), and (vi) of Theorem 6.3.1.

7. Prove theorem 6.4.2.

8. Using Theorems 6.4.1 and 6.4.2, give a proof of Theorem 6.4.3 (**The Kalman Canonical Decomposition Theorem**)

9. Prove that the change of variable $\bar{x} = Tx$, where T is nonsingular, preserves the controllability and observability of the system (A, B, C) .

10. Work out an algorithm to compute the nonsingular transforming matrix that transforms the pair (A, b) to the upper companion form. When can the matrix transforming T be highly ill-conditioned?

Construct a numerical example to support your statement.

11. Apply the test based on the eigenvalue criterion of controllability to Example 6.6.2 and show that this test will do better than the one based on the Kalman criterion.

12. Applying the stair-case algorithm to the pair (A, b) in Example 6.6.2, show that the pair (A, b) is uncontrollable.

13. If the controller-Hessenberg pair (H, \bar{B}) of the controllable system (A, B) is such that the subdiagonal blocks of H are nearly rank-deficient, then the system may be very near to an uncontrollable system.

- (a) Construct examples both in the single and multi-input cases in support of the above statement.
- (b) Construct another example to show that the converse is not necessarily true; that is, even the subdiagonal blocks of H have robust ranks, the system may be close to an uncontrollable system.
14. Show that to check the controllability for the pair (A, B) , where $A = \text{diag}(1, 2^{-1}, \dots, 2^{1-n})$ and $B = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$, the eigenvalue-criterion for controllability (the PBH criterion) will do better than the Kalman criterion.
15. Let $(H = (H_{ij}), \bar{B})$ be the controller-Hessenberg form of the pair (A, B) . Then prove the followings:
- If $H_{i,i-1} = 0$ for any i , then (A, B) is not controllable.
 - $\mu(A, B) \leq \min_{1 \leq i \leq k} \|H_{i,i-1}\|_2$.
16. Prove that $\mu(A, B)$ remains invariant under an orthogonal transformation.
17. Let (\tilde{A}, \tilde{b}) be as in (6.5.2). Prove that $\mu(\tilde{A}, \tilde{b}) \leq \sin\left(\frac{\pi}{n}\right)$ (Kenney and Laub (1988))
18. Develop an algorithm for the reduction of the pair (A, C) to the observer-Hessenberg form (6.8.1), without invoking the algorithm for the controller-Hessenberg reduction to the pair (A^T, C^T) . How can one obtain the observability indices of the pair (A, C) from this form?
19. Construct a simple example to show that the minimum which yields $\mu(A, B)$ is not achieved when s is an eigenvalue of A .
20. Rework Example 6.9.1 with the initial point as one of the eigenvalues of $F = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, where $[C, D]$ is a random matrix such that F is square.
21. Apply the Bisection method to Example 6.9.1 to find an estimate of $\mu_r(A, B)$.
22. Find $\mu(A, B)$ and $\mu_r(A, B)$, where A and B are given by

$$A = \begin{pmatrix} 0 & 1 & & & \\ & 0 & \ddots & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ & & & & 0 \end{pmatrix}_{10 \times 10} \quad B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}_{10 \times 1}$$

23. Derive Newton's algorithm for computing $\mu_r(A, B)$.

24. (Laub and Linnemann (1986). Consider the controllable pair

$$(H, \bar{b}) = \left(\begin{pmatrix} -4 & 0 & 0 & 0 \\ \alpha & -3 & 0 & 0 \\ 0 & \alpha & -2 & 0 \\ 0 & 0 & \alpha & -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right).$$

Show (experimentally or mathematically) that as $\alpha \rightarrow 0$, the pair (H, \bar{b}) becomes closer and closer to an uncontrollable pair.

25. Let (A, B) be controllable. Let $B = (B_1, B_2)$, with B_1 consisting of minimum number of inputs such that (A, B) is controllable.

Then prove that (A, B_1) is closer to an uncontrollable system than (A, B) is; that is, prove that

$$\min \sigma_n(B_1, A - sI) \leq \min \sigma_n(B, A - sI), s \in \mathbb{C}$$

26. (Gao and Neumann (1993)). Let $\lambda_0 \in \mathbb{C}$ and let $p \in \mathbb{C}$ be on the unit circle. Consider the straight line $\lambda = \lambda_0 + tp$, $t \in \mathbb{R}$. Then prove that

$$\min_{t \in \mathbb{R}} \sigma_{\min}(A - (\lambda_0 + tp)I, B) \leq \alpha$$

if and only if the matrix

$$G(\alpha) = \begin{pmatrix} \bar{p}(A - \lambda_0 I) & \bar{p}(BB^* - \alpha^2 I) \\ -pI & p(A^* - \bar{\lambda}_0 I) \end{pmatrix}$$

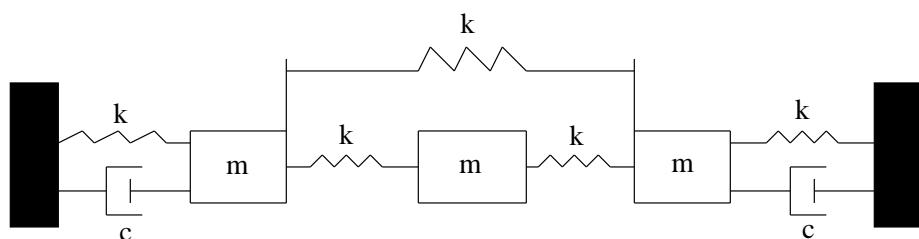
has a real eigenvalue.

Based on the above result, derive an algorithm for computing $\mu_R(A, B)$.

(Hint: take $\lambda_0 = 0$, $p = 1$).

Test your algorithm with Example 6.9.1.

27. (a) Construct a state-space representation of the following second-order model.



$$\begin{bmatrix} m & & \\ & m & \\ & & m \end{bmatrix} \begin{pmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_3 \end{pmatrix} + \begin{bmatrix} c & & \\ & 0 & \\ & & c \end{bmatrix} \begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{pmatrix} + \begin{bmatrix} 3k & -k & -k \\ -k & 2k & -k \\ -k & -k & 3k \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- (b) Show that the system is not controllable for

$$b = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, b = \begin{pmatrix} \alpha \\ \beta \\ \alpha \end{pmatrix} \text{ or } b = \begin{pmatrix} \beta \\ 0 \\ -\beta \end{pmatrix}.$$

28. Consider Example 5.2.6 on the motion of an orbiting satellite with $d_0 = 1$. Let $x(t) = (x_1(t), x_2(t), x_3(t), x_4(t))^T$, $u(t) = (u_1(t), u_2(t))^T$, and $y(t) = (y_1(t), y_2(t))^T$.

- (a) Show that one of the states cannot be controlled by the radial force $u_1(t)$ alone, but all the states can be controlled by using the tangential force $u_2(t)$.
(b) Show that all the states are observable using both the outputs; however, one of the states cannot be observed by $y_1(t)$ alone.

29. Let (H, \bar{b}) be the controller-Hessenberg pair of the controllable pair (A, b) such that $\| H \|_2 + \| \bar{b} \|_2 \leq \frac{1}{4}$. Then prove that the quantity $|\bar{b}_1 h_{21} h_{32} \cdots h_{n,n-1}|$ gives a lower bound on the perturbations needed to obtain an uncontrollable pair. Construct an example to support this.

30. Does the result of the preceding exercise hold in the multi-input case? Prove or disprove.

31. Consider the example of balancing a stick on your hand (Example 5.2.4). We know from our experience that a stick can be balanced. Verify this using the Kalman criterion of controllability. Take $L = 1$.

$$A = \begin{pmatrix} 0 & 1 \\ \frac{g}{L} & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ -\frac{g}{L} \end{pmatrix}.$$

32. **(An Uncontrollable System)** (Szidarovszky and Bahill (1991), pp. 223 - 224).

Consider the following electric network with two identical circuits in parallel. Intuitively, it is clear that there cannot exist a single input that will bring one circuit to one state and the other to a different state.

Verify this using the Kalman criterion of controllability. Take $L_1 = L_2 = 1$, $C_1 = C_2 = 1$, and $R_1 = R_2 = 1$.

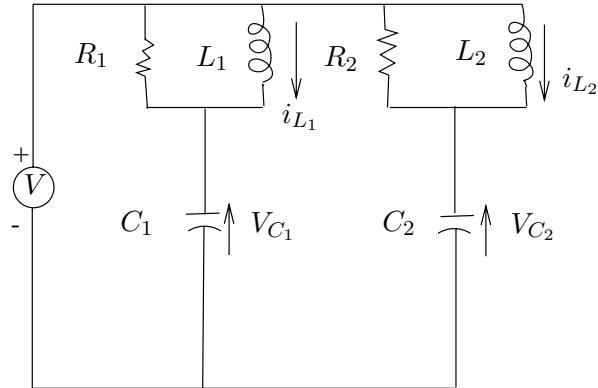


Figure 6.1 Uncontrollability of an Electrical Network

33. Consider Example 5.2.7 with $c = (0, 0, \dots, 0, 1)$. Show that the system is observable. How do you interpret this?

References

1. D. Boley, On Kalman's procedure for the computation of the controllable/observable canonical form, *SIAM J. Contr. Optimiz.*, vol. 18, pp. 624-626, 1980.
2. D. Boley, *Computing the Controllability/Observability of a Linear Time-Invariant Dynamic System: A Numerical Approach*, Ph.D. thesis, Comp. Sci. Dept., Stanford University, Stanford, CA, June 1981.
3. D. Boley and W.-S. Lu, Measuring how far a controllable system is from an uncontrollable system, *IEEE Trans. Automat. Control*, vol. AC-31, pp. 249-251, 1986.
4. D. Boley, Computing rank-deficiency of rectangular matrix pencils, *Syst. Contr. Lett.*, vol. 9, pp. 207-214, 1987.
5. D. Boley, Computing the Kalman decomposition, an optimal method, *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 51-53, 1984; correction in vol. 36, pp. 1341, 1991.
6. W. L. Brogan, *Modern Control Theory*, 3rd Edition, Prentice Hall, Englewood Cliffs, NJ, 1991.
7. R. Brockett, *Finite Dimensional Linear Systems*, Wiley, New York, 1970.
8. C.-T. Chen, *Linear Systems Theory and Design*, CBS College Publishing, New York, 1984.
9. R.A. DeCarlo, *Linear Systems: A State Variable Approach with Numerical Implementation*, Prentice Hall, Englewood Cliffs, NJ, 1989.
10. B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, Pacific Grove, CA, 1995.
11. R. Eising, Between controllable and uncontrollable, *Syst. Contr. Lett.*, vol. 4, pp. 263-264, 1984.
12. L. Elsner and C. He, An algorithm for computing the distance to uncontrollability, *Syst. Contr. Lett.*, vol. 17, pp. 453-464, 1991.
13. M. Gao and M. Neumann, A global minimum search algorithm for estimating the distance to uncontrollability, *Lin. Alg. Appl.*, vol. 188/189, pp. 305-350, 1993.
14. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Edition, Johns Hopkins University, Baltimore, MD, 1996.
15. M. L. J. Hautus, Controllability and observability conditions of linear autonomous systems, *Proc. Kon. Ned. Akad. Wetensch*, Ser. A. vol. 72, pp. 443-448, 1969.

16. R.A. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
17. T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.
18. R.E. Kalman, P.L. Flab and M.A. Arbib, *Topics in Mathematical Systems Theory*, McGraw Hill, New York, 1969.
19. C.S. Kenney and A.J. Laub, Controllability and stability radii for companion form systems, *Math. Control, Signals and Systems*, vol. 1, pp. 239-256, 1988.
20. A.J. Laub and A. Linnemann, Hessenberg forms in linear systems theory, *Computational and Combinatorial Methods in Systems Theory*, Elsevier Science Publishers, North Holland, pp. 229-244, 1986, (C.I. Byrnes and A. Lindquist, Editors).
21. A.J. Laub and A. Linnemann, Hessenberg and Hessenberg/triangular forms in linear systems theory, *Int. J. Control*, vol. 44, pp. 1523-1547, 1986.
22. D.G. Luenberger, *Introduction to Dynamic Systems: Theory, Models & Applications*, John Wiley & Sons, New York, 1979.
23. G.S. Miminis, *Numerical Algorithms for Controllability and Eigenvalue Allocations*, Master Thesis, School of Computer Science, McGill University, Montreal, Canada, May, 1981.
24. C.C. Paige, Properties of numerical algorithms related to computing controllability, *IEEE Trans. Automat. Control*, vol. AC-26, pp. 130-138, 1981.
25. M.M. Rosenbrock, *State-Space and Multivariable Theory*, John Wiley, New York, 1970.
26. F. Szidarovszky and A. Terry Bahill, *Linear Systems Theory*, CRC Press, Boca Raton, 1991.
27. P. M. Van Dooren and M. Verhaegen, On the use of unitary state-space transformations, Contemporary Mathematics, *Contemporary Mathematics Series*, Amer. Math. Soc., Providence, RI, vol. 47, pp. 447-463, 1985, (R. Brualdi, et al., Editors).
28. Varga, A., Numerically reliable algorithm to test controllability, *Electronic Lett.*, vol. 15, pp. 452-3, 1979.
29. M. Wicks and A. DeCarlo, Computing the distance to an uncontrollable system, *IEEE Trans. Automat. Control*, vol. 36, pp. 39-49, 1991.

Chapter 7

STABILITY, INERTIA, AND ROBUST STABILITY

Contents

7.1	Introduction	238
7.2	Stability of a Continuous-time System	239
7.2.1	Eigenvalue Criterion of Continuous-Time Stability	240
7.2.2	Continuous-time Lyapunov Stability Theory	241
7.2.3	Lyapunov Equations and Controllability and Observability Grammians	246
7.2.4	Lyapunov Equations and the H_2 -Norm	247
7.3	Stability of a Discrete-time System	249
7.3.1	Stability of a Homogeneous Discrete-Time System	249
7.4	Some Inertia Theorems	252
7.4.1	The Sylvester Law of Inertia	252
7.4.2	The Lyapunov Inertia Theorems	253
7.5	Determining the Stability and Inertia of a Nonsymmetric Matrix	256
7.6	Distance to an Unstable System	260
7.7	Robust Stability	266
7.8	The Structured Stability Radius	271
7.9	Some Selected Software	275
7.9.1	MATLAB CONTROL SYSTEM TOOLBOX	275
7.9.2	MATCONTROL	275
7.9.3	SLICOT	275
7.10	Summary and Review	275
7.11	Chapter Notes and Further Reading	277

Topics Covered

- Lyapunov Stability Theory for continuous-time and Discrete-time Systems.
- Controllability and Observability Grammians via Lyapunov Equations.
- Theory and Computation of the Inertia
- Distance to Instability
- Robust Stability
- Stability Radius

7.1 Introduction

This chapter deals with **stability of a linear time-invariant system** and the associated aspects such as the **inertia of a matrix**, **distance from an unstable system**, **robust stability**, and **stability radius** and computing the **H_2 -norm of a stable transfer function**. A classical approach to determine stability and inertia is to solve a **Lyapunov equation** or to find the characteristic polynomial of the state matrix A followed by application of the **Routh-Hurwitz criterion** in the continuous-time case and the **Schur-Cohn** criteria in the discrete-time case. These approaches are historically important and were developed at a time when numerically finding the eigenvalues of a matrix, even of a modest order, was a difficult problem. However, nowadays, with the availability of the QR iteration method for eigenvalue computation which is reliable, accurate and fast, these approaches for stability and inertia, seem to have very little practical value. Furthermore, the Lyapunov equation approach is counter-productive in a practical computational setting in the sense that the most numerically viable method, currently available for solution of the Lyapunov equation, namely, the **Schur method (described in Chapter 8)**, is based on transformation of the matrix A to a real Schur form and the later either explicitly displays the eigenvalues of A or the eigenvalues can be trivially found once A is transformed into this form. Also, as mentioned before, **finding the characteristic polynomial of a matrix, in general, is a numerically unstable process**. In view of the above statements, it is clear that the best way to numerically check the stability and inertia is to explicitly compute all the eigenvalues. However, by computing the eigenvalues, one gets more than stability and inertia. Furthermore, if the eigenvalues of A are very ill-conditioned, determining the stability and inertia using eigenvalues may be misleading (see **Section 7.6**). The question, therefore, arises if an approach can be developed that does not require explicit computation of the eigenvalues of the state matrix A nor solution of a

Lyapunov equation. Such an implicit method (**Algorithm 7.5.1**) is developed in **Section 7.5**. This method is about three times faster than the eigenvalue method and, of course, many times faster than solving Lyapunov equation in a numerically effective way using the Schur method.

An important practical problem “How nearly unstable is a stable system (or equivalently a stable matrix)?” is discussed in **Section 7.6**. A simple bisection algorithm due to Byers (1988) to measure the distance of a stable matrix A from a set of unstable matrices is provided. A brief discussion of robust stability is the topic of **Section 7.7**.

The concept of **stability radius** in the context of robust stability is introduced in **Section 7.8** and a recent important formula for real stability radius due to Qiu et al. (1995) is stated. This concept will again be revisited in **Chapter 10**, where a connection of the complex stability radius with an algebraic Riccati equation will be made.

The relationships between the controllability and observability Grammians and the H_2 -norm of an asymptotically stable system with Lyapunov equations are discussed in **Sections 7.2.3, 7.2.4 and 7.3**, and a computational algorithm (**Algorithm 7.2.1**) for computing the H_2 -norm of a stable continuous-time system is described in **Section 7.2.4**.

Reader’s Guide to Chapter 7

The readers familiar with the basic concepts of stability and Lyapunov stability theory can skip Sections 7.2 and 7.3.

7.2 Stability of a Continuous-time System

The stability of a system is defined with respect to an equilibrium state.

Definition 7.2.1 An **equilibrium state** of the unforced system

$$\dot{x}(t) = Ax(t), x(0) = x_0, \quad (7.2.1)$$

is the vector x_e satisfying

$$Ax_e = 0.$$

Clearly, $x_e = 0$ is an equilibrium state and it is the unique equilibrium state if and only if A is nonsingular.

Definition 7.2.2 An equilibrium state x_e is **asymptotically stable** if for any initial state, the state vector $x(t)$ approaches x_e as time increases.

The system (7.2.1) is asymptotically stable if and only if the equilibrium state $x_e = 0$ is asymptotically stable. Thus, the system (7.2.1) is asymptotically stable if and only if $x(t) \rightarrow 0$ as $t \rightarrow \infty$.

7.2.1 Eigenvalue Criterion of Continuous-Time Stability

Below we state a well-known criterion of asymptotic stability of a continuous-time system.

Theorem 7.2.1 *The system (7.2.1) is asymptotically stable if and only if all the eigenvalues of the matrix A have negative real parts.*

Proof: From Chapter 5, we know that the general solution of (7.2.1) is

$$x(t) = e^{At}x_0.$$

Thus $x(t) \rightarrow 0$ if and only if $e^{At} \rightarrow 0$ as $t \rightarrow \infty$. We will now show that this happens if and only if all the eigenvalues of A have negative real parts.

Let $X^{-1}AX = \text{diag}(J_1, J_2, \dots, J_k)$ be the Jordan canonical form of A . Then

$$e^{At} = X \text{diag}(e^{J_1 t}, e^{J_2 t}, \dots, e^{J_k t})X^{-1}.$$

Let λ_i be the eigenvalue of A associated with J_i . Then $e^{J_i t} \rightarrow 0$ if and only if λ_i has a negative real part. Therefore, $e^{At} \rightarrow 0$ if and only if all the eigenvalues of A have negative real parts.

Definition 7.2.3 *A matrix A is called a **stable matrix** if all of the eigenvalues of A have negative real parts.*

A stable matrix is also known as a **Hurwitz** matrix in control literature. In analogy, an eigenvalue with negative real part is called a **stable eigenvalue**.

Since the asymptotic stability of (7.2.1) implies that its zero-input response approaches zero exponentially, the asymptotic stability is also referred to as **exponential stability**.

Definition 7.2.4 *Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of A . Then the distance $\min\{-\text{Re}(\lambda_i) : i = 1, \dots, n\}$ to the imaginary axis is called the **stability margin** of A .*

In this book, the “stability” of a system means “asymptotic stability”; and, the associated matrix A will be referred to as “stable matrix”, not asymptotically stable matrix.

Bounded-Input Bounded-Output Stability

The continuous-time linear system

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t)\end{aligned}\tag{7.2.2}$$

is said to be **bounded-input bounded-output** (BIBO) **stable** if for any bounded input, the output is also bounded.

The BIBO stability is governed by the poles of the transfer function $G(s) = C(sI - A)^{-1}B$. Specifically, the following result can be proved: [Exercise 6].

Theorem 7.2.2 *The system (7.2.2) is BIBO stable if and only if every pole of $G(s)$ has a negative real part. ■*

Since every pole of $G(s)$ is also an eigenvalue of A , **an asymptotically stable system is also BIBO stable.** However, **the converse is not true.** The following simple example illustrates this.

Example 7.2.1

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}x + \begin{pmatrix} 0 \\ 1 \end{pmatrix}u, \\ y &= (1, 1)x.\end{aligned}$$

$$G(s) = C(sI - A)^{-1}B = (1, 1) \begin{pmatrix} s-1 & 0 \\ 0 & s+1 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{s+1}.$$

Thus the system is BIBO (note that the pole of $G(s)$ is -1), but not asymptotically stable.

Bounded-Input Bounded-State (BIBS) Stability.

Definition 7.2.5 *The system 7.2.2 is **bounded-input bounded-state (BIBS) stable** if, for any bounded input, the state response is also bounded.*

The following characterization of BIBS can be given in terms of eigenvalues of A and the controllability of the modes. For a proof of Theorem 7.2.2', see (DeCarlo (1989), pp. 416-417).

Theorem 7.2.2'. *(Bounded-Input Bounded-State Stability). The system (7.2.2) is bounded-input bounded-state stable if and only if*

- (i) All the eigenvalues of A have non-negative real parts.
- (ii) If an eigenvalue λ_i has a zero real part, then the order of the associated factor in the minimal polynomial of A must be 1.
- (iii) The mode associated with an eigenvalue with zero real part must be uncontrollable.

7.2.2 Continuous-time Lyapunov Stability Theory

In this section, we present the historical Lyapunov criterion of stability.

Before the advent of computers, finding the eigenvalues of a matrix A was an extremely difficult task. The early research on stability, therefore, was directed towards finding the criteria that do not require explicit computation of the eigenvalues of a matrix.

In 1892, the Russian mathematician A. Lyapunov developed a historical stability criterion for nonlinear systems of equations. In the linear case, this criterion may be formulated in terms of the solution of a matrix equation.

Theorem 7.2.3 (Lyapunov Stability Theorem)

The system

$$\dot{x}(t) = Ax(t)$$

is asymptotically stable if and only if, for any symmetric positive definite matrix M , there exists a unique symmetric positive definite matrix X satisfying the equation

$$XA + A^T X = -M. \quad (7.2.3)$$

Proof: Let's define a matrix X by

$$X = \int_0^\infty e^{A^T t} M e^{At} dt \quad (7.2.4)$$

Then, we show that when the system is asymptotic stable X is a unique solution of the equation (7.2.3) and is symmetric positive definite.

Using the expression of X in (7.2.3), we obtain

$$\begin{aligned} XA + A^T X &= \int_0^\infty e^{A^T t} M e^{At} Adt + \int_0^\infty A^T e^{A^T t} M e^{At} dt \\ &= \int_0^\infty \frac{d}{dt} (e^{A^T t} M e^{At}) dt = [e^{A^T t} M e^{At}]_0^\infty \end{aligned}$$

Since A is stable, $e^{A^T t} \rightarrow 0$ as $t \rightarrow \infty$. Thus $XA + A^T X = -M$, showing that X defined by (7.2.4) satisfies the equation (7.2.3).

To show that X is positive definite, we have to show that $u^T X u > 0$ for any nonzero vector u . Using (7.2.4) we can write

$$u^T X u = \int_0^\infty u^T e^{A^T t} M e^{At} u dt.$$

Since the exponential matrices $e^{A^T t}$ and e^{At} are both nonsingular and M is positive definite, we conclude that $u^T X u > 0$.

To prove that X is unique, assume that there are two solutions X_1 and X_2 of (7.2.3). Then

$$A^T(X_1 - X_2) + (X_1 - X_2)A = 0,$$

which implies that

$$e^{A^T t} (A^T(X_1 - X_2) + (X_1 - X_2)A) e^{At} = 0$$

or

$$\frac{d}{dt} [e^{A^T t} (X_1 - X_2) e^{At}] = 0,$$

and hence $e^{A^T t} (X_1 - X_2) e^{At}$ is a constant matrix for all t .

Evaluating at $t = 0$ and $t = \infty$ we conclude that $X_1 - X_2 = 0$.

We now prove the converse; that is, we prove that if X is a symmetric positive definite solution of the equation (7.2.3), then A is stable.

Let (λ, x) be an eigenpair of A . Then premultiplying the equation (7.2.3) by x^* and postmultiplying it by x , we obtain

$$x^* X A x + x^* A^T X x = \lambda x^* X x + \bar{\lambda} x^* X x = (\lambda + \bar{\lambda}) x^* X x = -x^* M x.$$

Since M and X are both symmetric positive definite, we have $\lambda + \bar{\lambda} < 0$ or $\operatorname{Re}(\lambda) < 0$.

■

Note: The matrix X defined by (7.2.4) satisfies the equation (7.2.3) even when M is not positive definite.

Definition 7.2.6 *The matrix equation*

$$XA + A^T X = -M$$

and its dual

$$AX + XA^T = -M$$

are called the **Lyapunov equations**.

Remark (Lyapunov Function): The Lyapunov stability theory was originally developed by Lyapunov in the context of stability of a nonlinear system. The stability of a nonlinear system is determined by **Lyapunov functions**. See Luenberger (1979) for details. For the linear system

$$\dot{x}(t) = Ax(t),$$

the function $V(x) = x^T X x$, where X is symmetric is a Lyapunov function if the $\dot{V}(x)$, the derivative of $V(x)$, is negative definite. This fact yields an alternative proof of Theorem 7.2.3. This can be seen as follows:

$$\begin{aligned}\dot{V}(x) &= \dot{x}^T X x + x^T \dot{X} \dot{x} \\ &= x^T (A^T X + X A) x \text{ (since } \dot{x} = Ax\text{)} \\ &= x^* (-M) x.\end{aligned}$$

Thus $\dot{V}(x)$ is negative definite if and only if M is positive definite.

We note the following from the proof of Theorem 7.2.3.

Integral Representations of the Unique Solutions of Lyapunov Equations

Let A be a stable matrix and let M be symmetric, positive definite or semidefinite. Then

- (i) The unique solution X of the Lyapunov equation

$$XA + A^T X = -M$$

is given by

$$X = \int_0^\infty e^{A^T t} M e^{At} dt \quad (7.2.5)$$

- (ii) The unique solution X of the Lyapunov equation

$$AX + XA^T = -M$$

is given by

$$X = \int_0^\infty e^{At} M e^{A^T t} dt \quad (7.2.6)$$

As we will see later, the Lyapunov equations also arise in many other important control theoretic applications. In many of these applications, the right-hand side matrix M is positive semi-definite, rather than positive definite. The typical examples are $M = BB^T$ or $M = C^TC$, where B and C are, respectively, the input and output matrices. The Lyapunov equations of the above types arise in finding **Grammians** of a stable system (see **Section 7.2.3**).

Theorem 7.2.4 *Let A be a stable matrix. Then the Lyapunov equation*

$$XA + A^T X = -C^T C \quad (7.2.7)$$

has a unique symmetric positive definite solution X if and only if (A, C) is observable.

Proof: We first show that the observability of (A, C) and stability of A imply that X is positive definite.

Since A is stable, by (7.2.5) the unique solution X of the equation (7.2.7) is given by

$$X = \int_0^\infty e^{A^T t} C^T C e^{At} dt.$$

If X is not positive definite, then there exists a nonzero vector x such that $Xx = 0$. In that case

$$\int_0^\infty \|Ce^{At}x\|^2 dt = 0;$$

this means that $Ce^{At}x = 0$. Evaluating $Ce^{At}x = 0$ and its successive derivatives at $t = 0$, we obtain $CA^i x = 0, i = 0, 1, \dots, n-1$. This gives $O_M x = 0$, where O_M is the observability matrix. Since (C, A) is observable, O_M has full rank, and this implies that $x = 0$, which is a contradiction.

Hence $Ce^{At}x \neq 0$, for every t . So, X is positive definite.

Next, we prove the converse. That is, we prove that the stability of A and definiteness of X imply that (A, C) is observable. The proof is again by contradiction.

Suppose (A, C) is not observable. Then, according to criterion (v) of Theorem 6.3.1, there is an eigenvector x of A such that $Cx = 0$. Let λ be the eigenvalue corresponding to the eigenvector x . Then from the equation

$$XA + A^T X = -C^T C,$$

we have $x^* X A x + x^* A^T X x = -x^* C^T C x$ or $(\lambda + \bar{\lambda})x^* X x = -\|Cx\|^2$.

So, $(\lambda + \bar{\lambda})x^* X x = 0$.

Since A is stable, $\lambda + \bar{\lambda} < 0$. Thus

$$x^* X x = 0.$$

But X is positive definite, so x must be a zero vector, which is a contradiction. ■

We next prove a necessary and sufficient condition of stability assuming that (A, C) is observable.

Theorem 7.2.5 *Let (A, C) be observable. Then A is stable if and only if there exists a unique symmetric positive definite solution matrix X satisfying the Lyapunov equation (7.2.7).*

Proof: We have already proved the theorem in one direction, that is, we have proved that if A is stable and (A, C) is observable, then the Lyapunov equation (7.2.7) has a unique symmetric positive definite solution X given by

$$X = \int_0^\infty e^{A^T t} C^T C e^{At} dt.$$

We now prove the other direction. Let (λ, x) be an eigen-pair of A . Then as before we have

$$(\lambda + \bar{\lambda})x^* X x = -\|Cx\|^2.$$

Since (A, C) is observable, $Cx \neq 0$; and since X is positive definite, $x^* X x > 0$. Hence $\lambda + \bar{\lambda} < 0$, which means that A is stable. ■

For the sake of convenience, we combine the results of Theorems 7.2.4 and 7.2.5 in the following Theorem (Theorem 7.2.6).

In the rest of this Chapter, for notational convenience, a symmetric positive definite (positive semidefinite) matrix X will be denoted by the symbol $X > 0 (\geq 0)$.

Theorem 7.2.6 Let X be a solution of the Lyapunov equation (7.2.7). Then the following hold:

- (i) If $X > 0$ and (A, C) is observable, then A is a stable matrix.
- (ii) If A is a stable matrix and (A, C) is observable, then $X > 0$
- (iii) If A is a stable matrix and $X > 0$, then (A, C) is observable.

Since observability is a dual concept of controllability, the following results can be immediately proved by duality of Theorems 7.2.4 and 7.2.5.

Theorem 7.2.7 Let A be a stable matrix. Then the Lyapunov equation

$$AX + XA^T = -BB^T \quad (7.2.8)$$

has a unique symmetric positive definite solution X if and only if (A, B) is controllable. ■

Theorem 7.2.8 Let (A, B) be controllable. Then A is stable if and only if there exists a unique symmetric positive definite X satisfying the Lyapunov equation (7.2.8). ■

Theorems 7.2.7 and 7.2.8 can also be combined, in a similar manner, as in Theorem 7.2.6, to obtain the following:

Theorem 7.2.9 Let X be a solution of the Lyapunov equation (7.2.8). Then the following hold:

- (i) If $X > 0$ and (A, B) is controllable, then A is a stable matrix
- (ii) If A is a stable matrix and (A, B) is controllable, then $X > 0$.
- (iii) If A is a stable matrix and $X > 0$, then (A, B) is controllable.

7.2.3 Lyapunov Equations and Controllability and Observability Grammians

Definition 7.2.7 Let A be a stable matrix. Then the matrix

$$C_G = \int_0^\infty e^{At} BB^T e^{A^T t} dt \quad (7.2.9)$$

is called the **controllability Grammian**, and the matrix

$$O_G = \int_0^\infty e^{A^T t} C^T C e^{At} dt \quad (7.2.10)$$

is called the **observability Grammian**.

In view of these definitions, Theorems 7.2.7 and 7.2.4 can be, respectively, restated as follows.

Theorem 7.2.10 (Controllability Grammian and the Lyapunov Equation)

Let A be a stable matrix. Then the controllability Grammian C_G satisfies the Lyapunov equation

$$AC_G + C_G A^T = -BB^T \quad (7.2.11)$$

and is symmetric positive definite if and only if (A, B) is controllable. ■

Theorem 7.2.11 (Observability Grammian and the Lyapunov Equation)

Let A be a stable matrix. Then the observability Grammian O_G satisfies the Lyapunov equation

$$O_G A + A^T O_G = -C^T C \quad (7.2.12)$$

and is symmetric positive definite if and only if (A, C) is observable. ■

Example 7.2.2 Let

$$A = \begin{pmatrix} -1 & -2 & -3 \\ 0 & -2 & -1 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

The controllability Grammian C_G obtained by solving the Lyapunov equation (using MATLAB command **lyap**) $AX + XA^T = -BB^T$ is

$$C_G = \begin{pmatrix} 0.2917 & 0.0417 & 0.0417 \\ 0.0417 & 0.1667 & 0.1667 \\ 0.0417 & 0.1667 & 0.1667 \end{pmatrix},$$

which is clearly singular. So, (A, B) is not controllable.

Verify: The singular values of the controllability matrix C_M are 25.6766, 0.8425, and 0.

7.2.4 Lyapunov Equations and the H_2 -Norm.

In this section, we show how the H_2 -norm of the transfer matrix of an asymptotically stable continuous-time system can be computed using Lyapunov equations.

Definition 7.2.8 The H_2 -norm of the transfer matrix $G(s)$ of an asymptotically stable continuous-time system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \quad (7.2.13)$$

denoted by $\|G\|_2$, is defined by

$$\|G\|_2 = \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} \text{Trace}(G(j\omega)^* G(j\omega)) d\omega \right)^{\frac{1}{2}} \quad (7.2.14)$$

Thus, the H_2 -norm measures the steady-state covariance of the output response $y = Gv$ to the white noise inputs v .

Computing the H_2 -Norm

By Parseval's theorem in complex analysis (Rudin (1966), 191), (7.2.14) can be written as

$$\|G(s)\|_2 = \left(\int_0^\infty \text{Trace}(h^T(t)h(t))dt \right)^{\frac{1}{2}},$$

where $h(t)$ is the impulse response matrix

$$h(t) = Ce^{At}B.$$

Thus,

$$\begin{aligned} \|G\|_2^2 &= \text{Trace} \left(B^T \left(\int_0^\infty e^{A^T t} C^T C e^{At} dt \right) B \right) \\ &= \text{Trace} (B^T O_G B), \end{aligned}$$

where O_G is the observability Grammian given by (7.2.10).

Similarly, we can show that

$$\|G\|_2^2 = \text{Trace} (CC_G C^T), \quad (7.2.15)$$

where C_G is the controllability Grammian given by (7.2.9).

Since A is stable, the controllability and observability Grammians satisfy, respectively, the Lyapunov equations (Theorems 7.2.10 and 7.2.11):

$$AC_G + C_G A^T + BB^T = 0, \quad (7.2.16)$$

$$A^T O_G + O_G A + C^T C = 0 \quad (7.2.17)$$

Thus, a straightforward method for computing the H_2 -norm is as follows:

Algorithm 7.2.1 Computing the H_2 -norm.

Input: The system matrices A , B , and C .

Output: The H_2 -norm of the system (A, B, C) .

Assumption: A is stable.

Step 1. Solve the Lyapunov equation

$$AC_G + C_G A^T + BB^T = 0$$

or

$$A^T O_G + O_G A + C^T C = 0$$

Step 2. Compute either $\text{Trace}(CC_G C^T)$ or $\text{Trace}(B^T O_G B)$, depending upon which of the two Lyapunov equations is solved, and take either of these two values as the value of the H_2 -norm.

Example 7.2.3

$$A = \begin{pmatrix} -1 & 2 & 3 \\ 0 & -2 & 2 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Step 1. The solution of the Lyapunov equation (7.2.16), C_G , is

$$C_G = \begin{pmatrix} 9.1833 & 2.5667 & 1.0167 \\ 2.5667 & 1.0333 & 0.5333 \\ 1.0167 & 0.5333 & 0.3333 \end{pmatrix}.$$

$$C' = CC_GC^T = \begin{pmatrix} 18.7833 & 18.7833 \\ 18.7833 & 18.7833 \end{pmatrix}.$$

The solution of the Lyapunov equations (7.2.17), O_G , is

$$O_G = \begin{pmatrix} 1 & 1.3333 & 1.9167 \\ 1.3333 & 1.8333 & 2.7000 \\ 1.9167 & 2.7000 & 4.0500 \end{pmatrix}$$

$$B' = B^T O_G B = \begin{pmatrix} 18.7833 & 18.7833 \\ 18.7833 & 18.7833 \end{pmatrix}$$

Step 2. H_2 -norm = Trace(B') = Trace (C') = 37.5667.

MATCONTROL NOTE. Algorithm 7.2.1 has been implemented in MATCONTROL function **h2nrmcq** and **h2nrmog**.

7.3 Stability of a Discrete-time System

7.3.1 Stability of a Homogeneous Discrete-Time System

Consider the discrete-time system

$$x_{k+1} = Ax_k, \tag{7.3.1}$$

with initial value x_0 .

A well-known mathematical criterion for asymptotic stability of the homogeneous discrete-time system now follows. The proof is analogous to that of Theorem 7.2.1 and can be found in Datta (1995).

Theorem 7.3.1 *The system (7.3.1) is asymptotically stable if and only if all the eigenvalues of A are inside the unit circle.*

Definition 7.3.1 *A matrix A having all its eigenvalues inside the unit circle is called a **discrete-stable matrix**, or a **convergent matrix** or a **Schur matrix**.*

We shall use the terminology **discrete-stable** throughout the book.

Discrete-time Lyapunov Stability Theory

Each of the theorems in Section 7.2 has a discrete counterpart. In the discrete case, the continuous-time Lyapunov equations $XA + A^T X = -M$ and $AX + XA^T = -M$, are, respectively, replaced by their discrete-analogs $X - A^T X A = M$ and $X - A X A^T = M$.

These discrete counterparts of the continuous-time Lyapunov equations are called the **Stein equations**. The Stein equations are also known as **discrete-Lyapunov equations** in control literature.

In the following we state and prove a discrete analog of Theorem 7.2.1. The statements and proofs of the discrete counterparts of Theorems 7.2.3, through 7.2.9 are analogous. In fact, the Lyapunov and Stein equations are related via the matrix analogs of the well-known bilinear transformation (known as the **Cayley transformation**):

$$z = \frac{1+s}{1-s}, \quad s = \frac{z-1}{z+1} \quad (7.3.2)$$

Note that $|z| < 1 \Leftrightarrow \operatorname{Re}(s) < 0$ and $|z| = 1 \Leftrightarrow \operatorname{Re}(s) = 0$.

Theorem 7.3.2 (Discrete-time Lyapunov Stability Theorem)

The discrete-time system (7.3.1) is asymptotically stable if and only if, for any positive definite matrix M , there exists a unique positive definite matrix X satisfying the discrete Lyapunov equation

$$X - A^T X A = M. \quad (7.3.3)$$

Proof: We prove the Theorem in one direction, that is, we prove that if A is discrete-stable, then the equation (7.3.3) has a unique symmetric positive definite solution X . The proof of the other direction is left as an [Exercise 5].

Define the matrix

$$X = \sum_{k=0}^{\infty} (A^T)^k M A^k. \quad (7.3.4)$$

Since A is discrete-stable, the infinite series on the right-hand side converges. Furthermore, the matrix X is symmetric and positive definite.

We now show that X is the unique solution of the Equation (7.3.3). Indeed,

$$X - A^T X A = \sum_{k=0}^{\infty} (A^T)^k M A^k - \sum_{k=1}^{\infty} (A^T)^k M A^k = M. \quad (7.3.5)$$

Thus X defined by (7.3.4) satisfies the equation (7.3.3).

To prove that X is unique, let's assume that there is another symmetric positive definite solution X_1 of (7.3.3).

Then

$$X_1 - A^T X_1 A = M,$$

and

$$\begin{aligned} X &= \sum_{k=0}^{\infty} (A^T)^k M A^k = \sum_{k=0}^{\infty} (A^T)^k (X_1 - A^T X_1 A) A^k \\ &= \sum_{k=0}^{\infty} (A^T)^k X_1 A^k - \sum_{k=1}^{\infty} (A^T)^k X_1 A^k = X_1. \end{aligned}$$

■

Remarks. (BIBO and BIBS Stability of a Discrete-time System). Results on BIBO stability and BIBS stability, the discrete counter parts of Theorem 7.2.2 and Theorem 7.2.2' can be obtained, for the discrete-time system

$$x_{k+1} = Ax_k + Bu_k.$$

See **Exercise 7** and **Exercise 8** and the book by DeCarlo (1989).

Definition 7.3.2 Let A be discrete-stable. Then the matrices

$$C_G^D = \sum_{k=0}^{\infty} A^k B B^T (A^T)^k \quad (7.3.6)$$

and

$$O_G^D = \sum_{k=0}^{\infty} (A^T)^k C^T C A^k \quad (7.3.7)$$

are, respectively, called the **discrete-time controllability Grammian** and **discrete-time observability Grammians**.

The discrete counterparts of Theorems 7.2.10 and 7.2.11 are:

Theorem 7.3.3 (Discrete-time Controllability Grammian and Lyapunov Equation).

Let A be discrete-stable. Then the discrete-time controllability Grammian C_G^D satisfies the discrete Lyapunov equation

$$C_G^D - A C_G^D A^T = B B^T \quad (7.3.8)$$

and is symmetric positive definite if and only if (A, B) is controllable. ■

Theorem 7.3.4 (Discrete-time Observability Grammian and Lyapunov equation).

Let A be discrete-stable. Then the discrete-time observability Grammian O_G^D satisfies the discrete Lyapunov equation

$$O_G^D - A^T O_G^D A = C^T C \quad (7.3.9)$$

and is symmetric positive definite if and only if (A, C) is observable. ■

7.4 Some Inertia Theorems

As noted in Introduction, certain design specifications require that the eigenvalues lie in a certain region of the complex plane. Thus finding if a matrix is stable is not enough in many practical instances. Thus, we should consider the following more general problem, known as the inertia problem.

Definition 7.4.1 *The inertia of a matrix A of order n , denoted by $In(A)$, is the triplet $(\pi(A), \nu(A), \delta(A))$, where $\pi(A)$, $\nu(A)$, and $\delta(A)$ are, respectively, the number of eigenvalues of A with positive, negative, and zero real parts, counting multiplicities.*

Note that $\pi(A) + \nu(A) + \delta(A) = n$ and A is a **stable matrix if and only if** $In(A) = (0, n, 0)$.

The inertia, as defined above, is the **half-plane** or the **continuous-time inertia**.

The inertia with respect to the other regions of the complex plane can similarly be defined.

The **discrete-time inertia** or the **unit-circle inertia** is defined by the triplet $(\pi_O(A), \nu_O(A), \delta_O(A))$, where $\pi_O(A)$, $\nu_O(A)$, $\delta_O(A)$, are, respectively the number of eigenvalues of A outside, inside, and on the unit circle. It will be denoted by $In_O(A)$.

Unless otherwise stated, by “inertia” we will mean the “half-plane inertia.”

Much work has been done on the inertia theory of matrices. We will just give here a glimpse of the existing inertia theory and then present a computational algorithm for computing the inertia. For details, we refer the curious readers to the recent survey paper of the author (Datta (1999)). This paper gives an overview of the state-of-the-art theory and applications of matrix inertia and stability. The applications include new matrix theoretic proofs of several classical stability tests, applications to D -stability and to continued functions, etc. (Datta (1978a, 1978b, 1979, 1980). For other control-theoretic applications of the inertia of a matrix, see Glover (1984), and the book by Zhou et al. (1996).

7.4.1 The Sylvester Law of Inertia

A classical law on the inertia of a symmetric matrix A is the **Sylvester Law of Inertia**, stated as follows:

Let A be a symmetric matrix and P be a nonsingular matrix. Then

$$In(A) = In(PAP^T).$$

Proof. See Horn and Johnson (1985, 223-229).

Computing the Inertia of a Symmetric Matrix

If A is symmetric, then Sylvester’s law of inertia provides an inexpensive and numerically effective method for computing its inertia.

A symmetric matrix A admits a triangular factorization:

$$A = UDU^T$$

where U is a product of elementary unit upper triangular and permutation matrices, and D is a symmetric block diagonal with blocks of order 1 or 2. This is known as **diagonal pivoting factorization**. Thus, by Sylvester's law of inertia $In(A) = In(D)$. Once this diagonal pivoting factorization is obtained, the inertia of the symmetric matrix A can be obtained from the entries of D as follows:

Let D have p blocks of order 1 and q blocks of order 2, with $p + 2q = n$. Assume that none of the 2×2 blocks of D is singular. Suppose that out of p blocks of order 1, p' of them are positive, p'' of them are negative, and p''' of them are zero (that is, $p' + p'' + p''' = p$). Then

$$\begin{aligned}\pi(A) &= p' + q \\ v(A) &= p'' + q \\ \delta(A) &= p'''\end{aligned}$$

The **diagonal pivoting factorization can be achieved in a numerically stable way**. It requires only $n^3/3$ flops. For details of the diagonal pivoting factorization, see Bunch (1971), Bunch and Parlett (1971) and Bunch and Kaufman (1977).

LAPACK Implementation: The diagonal pivoting method has been implemented in the LAPACK routine **SSYTRF**.

7.4.2 The Lyapunov Inertia Theorems

The Sylvester Law of Inertia and the matrix formulation of the Lyapunov criterion of stability seem to have made a significant impact on the development of nonsymmetric inertia theorems. Many inertia theorems for nonsymmetric matrices have been developed over the years. These theorems attempt to find a symmetric matrix X , given a nonsymmetric matrix A , as a solution of a certain matrix equation, in such a way that, under certain conditions, the inertia of the nonsymmetric matrix A becomes equal to the inertia of the symmetric matrix X . Once the symmetric matrix X is obtained, its inertia can be computed rather cheaply by application of the Sylvester Law of Inertia to the LDL^T decomposition of X .

The following is the **Fundamental Theorem** on the inertia of a nonsymmetric matrix. The Theorem is known as the **Main Inertia Theorem (MIT)**.

Theorem 7.4.1 (The Main Inertia Theorem). (i) *There exists a unique symmetric matrix X such that*

$$XA + A^T X = M > 0 \tag{7.4.1}$$

if and only if $\delta(A) = 0$.

(ii) *Whenever the equation (7.4.1) has a symmetric solution X , $In(A) = In(X)$.* ■

Recovery of the Lyapunov Stability Theorem

As an immediate corollary of Theorem 7.4.1, we obtain the following.

Corollary 7.4.1 *A necessary and sufficient condition for A to be stable is that there exists a symmetric positive definite matrix X such that*

$$XA + A^T X = -M,$$

where $M > 0$. ■

The Lyapunov Stability Theorem (**Theorem 7.2.3**) now follows from Corollary 7.4.1 by noting the fact that the Lyapunov equation

$$XA + A^T X = -M,$$

for any given positive definite matrix M , has a unique solution if and only if $\Delta(A) = \prod_{i,j=1}^n (\lambda_i + \lambda_j) \neq 0$ where $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of A , and $\Delta(A) \neq 0$ implies that $\delta(A) = 0$ (see Chapter 8).

Remark: Theorem 7.4.1 as it appears above is due to Ostrowski and Schneider (1962). Part (ii) was proved by Taussky (1961) for the special case when X is a unique solution of the Lyapunov equation.

The following discrete-version of the MIT was mentioned as a “remark” by Taussky (1964). Wimmer (1973) gave a complete proof of the theorem.

Theorem 7.4.2 *There exists a unique symmetric matrix X such that*

$$A^T X A - X = M > 0$$

if and only if $\delta_O(A) = 0$. In this case $In_O(A) = In(X)$. ■

While the main inertia theorem and its discrete counterpart generalize the Lyapunov and Stein stability theorems (Theorems 7.2.3 and 7.3.2) and are important in their own rights, their uses are restricted. As we have seen before, and will see again later, that many practical control-theoretic applications give rise to the Lyapunov matrix equations with positive semidefinite matrices on the right hand sides.

The next theorem proved by Carlson and Schneider (1963) deals with a positive semidefinite matrix on the right hand side.

Theorem 7.4.3 (Continuous-Time Semidefinite Inertia Theorem). *Assume that $\delta(A) = 0$ and let X be a nonsingular symmetric matrix such that*

$$XA + A^T X = M \geq 0.$$

Then $In(A) = In(X)$. ■

The discrete version of the above theorem appears in Datta (1980).

Theorem 7.4.4 (Discrete-Time Semidefinite Inertia Theorem). *Let $\delta_O(A) = 0$ and let X be a nonsingular symmetric matrix such that*

$$A^T X A - X = M \geq 0.$$

Then $In_O(A) = In(X)$ ■

The applicability of Theorem 7.4.3 requires that $\delta(A) = 0$. Similarly, applicability of Theorem 7.4.4 requires that $\delta_0(A) = O$. However, it was observed independently by Chen (1973) and Wimmer (1974) that the condition $\delta(A) = 0$ in Theorem 7.4.3 can be replaced by the condition of the controllability of the pair (A, M) .

Similarly, Wimmer and Ziebur (1975) observed that the condition $\delta_O(A) = 0$ in Theorem 7.4.4 (that is, A does not have an eigenvalue of modulus 1) can be replaced by the controllability of the pair (A^T, M) . We will refer Theorem 7.4.5 and Theorem 7.4.6 as the Chen-Wimmer and Wimmer-Ziebur Theorem, respectively.

We now state and prove the Chen-Wimmer theorem for the continuous-time inertia and just state the Wimmer-Ziebur theorem for the discrete-time inertia.

Theorem 7.4.5 *Let X be a nonsingular symmetric matrix such that*

$$XA + A^T X = M \geq 0,$$

and let (A^T, M) be controllable. Then (i) $\delta(A) = 0$, and (ii) $In(A) = In(X)$.

Proof: Suppose that (A^T, M) is controllable but $\delta(A) \neq 0$. Since $\delta(A) \neq 0$, there is an eigenvalue λ of A such that $\lambda + \bar{\lambda} = 0$. Let x be an eigenvector of A corresponding to $\bar{\lambda}$. Then $x^* M x = x^*(XA + A^T X)x = (\bar{\lambda} + \lambda)x^* X x = 0$.

This is, according to the eigenvector-criterion of controllability in Chapter 6, in contradiction to the assumption that (A^T, M) is controllable. This proves part (i).

Part (ii) now follows from the Theorem 7.4.3. ■

Remark: We stated Theorem 7.4.5 in a way so that uniformity with the statement of Theorem 7.4.3 is maintained. **However, the assumption that X is nonsingular is not needed.** It can be shown [Exercise 14] that the controllability of (A^T, M) implies that $\delta(A) = \delta(X) = 0$. The discrete-version of the above result was obtained by Wimmer and Ziebur (1975).

Theorem 7.4.6 *Let X be a nonsingular symmetric matrix such that $A^T X A - X = M > 0$, and let (A^T, M) be controllable. Then A has no eigenvalue of modulus 1 and the number of eigenvalues of A with modulus less (greater) than 1 is equal to $\nu(X)$ ($\pi(X)$).*

Remark. It should be noted that **Theorems 7.4.1-7.4.6** are not valid for any symmetric positive definite or positive semidefinite matrix M on the right-hand side, but only for those M for which the Lyapunov equations admit symmetric solutions X . On the other hand, the Lyapunov stability theorem (**Theorem 7.2.3**) applies for any symmetric positive definite matrix M .

7.5 Determining the Stability and Inertia of a Nonsymmetric Matrix

From our discussions in the two previous sections, it is clear that the stability and inertia of a nonsymmetric matrix can be determined by solving an appropriate Lyapunov equation.

Unfortunately **this is computationally a counterproductive approach**. The reason is that the most numerically effective (and widely used) method for solving a Lyapunov equation, the Schur method (see **Chapter 8**), is based on reduction of the matrix A to the real Schur form. The real Schur form either displays the eigenvalues of A or can be trivially obtained from there. Of course, once the eigenvalues are computed, the stability and inertia are immediately known.

An alternative classical approach (see Marden (1966)) is to compute the characteristic polynomial of A , followed by application of the **Routh-Hurwitz criterion** in the continuous-time case and the **Schur-Cohn Criterion** in the discrete-time case. **This is, unfortunately, also not a numerically viable approach.** The reasons are that (i) computing the characteristic polynomial may be a highly numerically unstable process and (ii) the coefficients of a polynomial may be extremely sensitive to small perturbations. See our discussions in Chapter 4 (**Section 4.2**).

In view of the above considerations, the numerical analysts believe that the most numerically effective way to compute the inertia and stability of a matrix A is to explicitly compute the eigenvalues of A . However, by explicitly computing the eigenvalues of A , one gets more than what is needed, and furthermore, since the eigenvalues of a matrix can be sensitive to small perturbations, computing the inertia and stability this way may be quite misleading sometimes (see the example in *Section 7.6 which shows that a perfectly stable matrix becomes unstable by a very small perturbation of a single entry of the matrix*).

It is, therefore, of interest to develop a method for inertia and stability that does not require solution of a Lyapunov equation, or computation of the characteristic polynomial or the eigenvalues of A . We will now describe such a method.

The algorithm described below (Algorithm 7.5.1) is based on the **implicit solution** of a matrix equation. The algorithm constructs a symmetric matrix F which satisfies a Lyapunov matrix equation with a positive semidefinite matrix on the right hand side; but **the Lyapunov matrix equation is not explicitly solved**. The algorithm is due to Carlson and Datta (1979b).

Algorithm 7.5.1 An Implicit Matrix Equation Method for Inertia and Stability

Input: An $n \times n$ real matrix A

Output: The inertia of A .

Step 1. Transform A to a lower Hessenberg matrix H using an orthogonal similarity. Assume that H is unreduced.

Step 2. Construct a nonsingular lower triangular matrix L such that

$$LH + HL = R = \begin{pmatrix} 0 \\ r \end{pmatrix}$$

is a matrix whose first $(n - 1)$ rows are zero, starting with the first row l_1 of L as $l_1 = (1, 0, \dots, 0)$.

Step 3. Having constructed L , compute the last row r of R .

Step 4. Construct now a matrix S such that

$$SH = H^T S,$$

with the last row s_n of S as the last row r of R .

Step 5. Compute $F = L^T S$.

Step 6. If F is nonsingular, compute the inertia of the symmetric matrix F , using the Sylvester law of inertia, as described in Section 7.4.1.

Step 7. Obtain the inertia of A : $\text{In}(A) = \text{In}(F)$.

Theorem 7.5.1 (i) If F is nonsingular, then it is symmetric and $\text{In}(A) = \text{In}(F)$.

(ii) A is stable if and only if F is negative definite.

Proof: *Proof of Part (i).*

$$\begin{aligned} FH + H^T F &= L^T SH + H^T L^T S \\ &= L^T H^T S + H^T L^T S \\ &= (L^T H^T + H^T L^T)S \\ &= R^T S = r^T r \geq 0. \end{aligned}$$

The nonsingularity of F implies the nonsingularity of S , and it can be shown [see Datta and Datta (1987)] that S is nonsingular if and only if H and $-H$ do not have a common eigenvalue. Thus, F is a unique solution of the matrix equation (see Theorem 8.2.1):

$$FH + H^T F = r^T r \geq 0,$$

and is, therefore, necessarily symmetric. Furthermore, since H and $-H$ do not have a common eigenvalue, we have $\delta(H) = 0$. Theorem 7.4.3 now can be applied to the above matrix equation to obtain Theorem 7.5.1.

Proof of Part (ii). First suppose that A is stable, then we prove that F is negative definite.

Since A is stable, so is H , and therefore, $\delta(H) = 0$. Again $\delta(H) = 0$ implies that H and $-H$ do not have an eigenvalue in common. Therefore by Theorem 8.2.1 (see Chapter 8), the Lyapunov equation

$$FH + H^T F = r^T r \geq 0$$

has a unique solution F and therefore, must be symmetric F . By Theorem 7.4.3, we then have

$$\text{In}(F) = \text{In}(A) = (0, n, 0).$$

Thus F is negative definite. Conversely, let F be negative definite. Then F is nonsingular. By part (i), we then have that $\text{In}(A) = \text{In}(F) = (0, n, 0)$. So, A is stable.

Computational Remarks:

Computation of L . Once the first row of $L = (l_{ij})$ in step 2 is prescribed, the diagonal entries of L are immediately known. These are: $1, -1, 1, \dots, (-1)^{n-1}$. Having known these diagonal entries, the $\frac{n(n-1)}{2}$ off-diagonal entries $l_{ij} (i > j)$ of L lying below the main diagonal can now be uniquely determined by solving a lower triangular system if these entries are computed in the following order: $l_{21}, l_{31}, l_{32}, \dots, l_{n1}, l_{n2}, \dots, l_{n,n-1}$.

Computation of S . Similar remarks hold for computing S in step 4. Knowing the last row of the matrix S , the rows s_{n-1} through s_1 of S can be computed directly from the relation $SH = H^T S$.

Notes:

1. The above algorithm has been modified and made more efficient by Datta and Datta (1987). The modified algorithm uses the matrix-adaptation of the well-known Hyman method for computing the characteristic polynomial of a Hessenberg matrix [see Wilkinson (1965)], which is numerically effective with proper scaling.
2. The algorithm has been extended by Datta and Datta (1986) to obtain information on the number of eigenvalues of a matrix in several other regions of the complex plane including strips, ellipses, and parabolas.
3. A method of this type for finding distribution of eigenvalues of a matrix with respect to the unit circle has been reported by L. Z. Lu (an unpublished manuscript (1987)).
4. A comparison of various methods for inertia computation, and a computationally more effective version of the algorithm reported in this section appeared in the M. Sc. Thesis of Daniel Pierce (1983).

The semidefinite inertia theorems of Section 4 play an important role in deriving these methods.

Flop-Count of Algorithm 7.5.1 and Comparisons with other Methods:

The algorithm requires about n^3 flops once the matrix A has been transformed to the lower Hessenberg matrix H . Since it requires $\frac{10}{3}n^3$ flops to transform A to H , a total of about $\frac{13}{3}n^3$

flops is needed to determine the inertia and stability of A using Algorithm 7.5.1. This count compares very favorable with about $12n^3$ flops needed to compute the eigenvalues of A using the QR iteration algorithm described in Chapter 4. **Thus Algorithm 7.5.1 is about 3 times faster than the eigenvalue method.**

We have not included the Lyapunov equation approach and the characteristic polynomial approach in our comparisons here, because of the numerical difficulties with the characteristic polynomial approach and the counterproductivity of the Lyapunov equation approach, as mentioned in the beginning of this section.

Example 7.5.1 We compute $\text{In}(A)$ using Algorithm 7.5.1 with

$$A = \begin{pmatrix} 1.997 & -0.724 & 0.804 & -1.244 & -1.365 & -2.014 \\ 0.748 & 2.217 & -0.305 & 1.002 & -2.491 & -0.660 \\ -1.133 & -1.225 & -0.395 & -0.620 & 1.504 & 1.498 \\ -0.350 & 0.515 & -0.063 & 2.564 & 0.627 & 0.422 \\ -0.057 & -0.631 & 1.544 & 0.001 & 1.074 & -1.750 \\ -1.425 & -0.788 & 1.470 & -1.515 & 0.552 & -0.036 \end{pmatrix}$$

Step 1: Reduction to Lower Hessenberg form:

$$H = \begin{pmatrix} 1.9970 & 2.9390 & & & & \\ 0.6570 & -1.0007 & 1.9519 & & & \\ 0.4272 & 1.5242 & 0.4502 & 0.8785 & & \\ -0.1321 & 1.2962 & 0.9555 & 1.4541 & 0.4940 & \\ -0.0391 & -1.5738 & 0.6601 & 0.2377 & 2.3530 & -0.4801 \\ -1.8348 & -0.5976 & 0.7595 & 0.1120 & -3.3993 & 2.1673 \end{pmatrix}$$

Step 2: Construction of the lower triangular L such that $LH + HL = R$:

$$L = \begin{pmatrix} 1 & & & & & \\ -1.3590 & 1 & & & & \\ 0.6937 & 1.0209 & 1 & & & \\ -1.3105 & -1.6810 & -3.2933 & 1 & & \\ 16.4617 & 22.7729 & 19.3373 & 11.7433 & 1 & \\ 198.8687 & 258.5635 & 229.9657 & 128.4966 & 21.8842 & 1 \end{pmatrix}$$

Step 3: Last row of the matrix R is

$$r = (1023.6330, 1293.0942, 1177.7393, 632.4162, 162.4031, -14.8420)$$

Step 4: Construction of S such that $SH = H^T S$:

$$S = \begin{pmatrix} 2.1404 & 3.3084 & 2.7775 & 1.3224 & -1.0912 & 0.4808 \\ 3.3084 & 5.0426 & 4.1691 & 2.0997 & -1.2521 & 0.6073 \\ 2.7775 & 4.1691 & 3.6050 & 1.8169 & -1.1757 & 0.5531 \\ 1.3224 & 2.0996 & 1.8169 & 0.8899 & -0.6070 & 0.2970 \\ -1.0912 & -1.2521 & -1.1757 & -0.6070 & -0.3845 & 0.0763 \\ 0.4808 & 0.6073 & 0.5531 & 0.2970 & 0.0763 & -0.0070 \end{pmatrix}$$

Step 5: Computation of $F = L^T S$:

$$F = \begin{pmatrix} 75.4820 & 96.7596 & 87.8785 & 47.6385 & 9.4298 & -0.4808 \\ 96.7596 & 124.1984 & 112.7028 & 61.2339 & 12.0384 & -0.6073 \\ 87.8785 & 112.7028 & 102.0882 & 55.4523 & 10.9292 & -0.5531 \\ 47.6385 & 61.2339 & 55.4523 & 30.1476 & 5.8930 & -0.2970 \\ 9.4298 & 12.0384 & 10.9292 & 5.8930 & 1.2847 & -0.0763 \\ -0.4808 & -0.6073 & -0.5531 & -0.2970 & -0.0763 & 0.0070 \end{pmatrix}$$

Step 6: Gaussian Elimination with diagonal pivoting: $PFP^T = WDW^T$, gives

$$W = \begin{pmatrix} 1 & & & & & \\ 0.9074 & 1 & & & & \\ 0.7791 & -0.4084 & 1 & & & \\ 0.0969 & -0.0274 & 0.4082 & 1 & & \\ 0.4930 & 0.6227 & -0.8765 & 0.0102 & 1 & \\ -0.0049 & 0.0111 & -0.0651 & -0.1454 & 0.0502 & 1 \end{pmatrix},$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$D = \begin{pmatrix} 124.1984 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.1831 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1298 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0964 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.0715 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0016 \end{pmatrix}.$$

Step 7. $\text{In}(A) = \text{In}(F) = \text{In}(D) = (4, 2, 0)$.

Verification: The eigenvalues of A are:

$$\{-2.1502, 0.8553, 3.6006, 2.0971, 3.1305, -0.1123\},$$

confirming that $\text{In}(A) = (4, 2, 0)$.

MATCONTROL NOTE. Algorithm 7.5.1 has been implemented in MATCONTROL function **inertia**.

7.6 Distance to an Unstable System

Let A be an $n \times n$ complex stable matrix. A natural question arises:

How “nearly unstable” is the stable matrix A ?

We consider the above question in this section.

Definition 7.6.1 Let $A \in \mathbb{C}^{n \times n}$ have no eigenvalue on the imaginary axis. Let $U \in \mathbb{C}^{n \times n}$ be the set of matrices having at least one eigenvalue on the imaginary axis. Then, with $\|\cdot\|$ as the 2-norm or the Frobenius norm, the distance from A to U is defined by

$$\beta(A) = \min\{\|E\| \mid A + E \in U\}.$$

If A is stable, then $\beta(A)$ is the distance to the set of unstable matrices.

The concept of “distance to instability” is an important practical concept. Note that a theoretically perfect stable matrix may be very close to an unstable matrix. For example, consider the following matrix (Petkov et al. (1991)):

$$A = \begin{pmatrix} -0.5 & 1 & 1 & 1 & 1 & 1 \\ 0 & -0.5 & 1 & 1 & 1 & 1 \\ 0 & 0 & -0.5 & 1 & 1 & 1 \\ 0 & 0 & 0 & -0.5 & 1 & 1 \\ 0 & 0 & 0 & 0 & -0.5 & 1 \\ 0 & 0 & 0 & 0 & 0 & -0.5 \end{pmatrix}$$

Since its eigenvalues are all -0.5 , it is perfectly stable. However, if the $(6, 1)$ th entry is perturbed to $\epsilon = \frac{1}{324}$ from zero, then the eigenvalues of this slightly perturbed matrix become: $-0.8006, -0.7222 \pm 0.2485j, -0.3775 \pm 0.4120j, 0.000$.

Thus, the perturbed matrix is unstable, showing that the stable matrix A is very closed to an unstable matrix.

We now introduce a measure of $\beta(A)$ in terms of singular values and describe a simple bisection algorithm to approximately measure it.

Let $\sigma_{\min}(A - j\omega I)$ be the smallest singular value of $A - j\omega I$. Then it can be shown [Exercise 15] that

$$\beta(A) = \min_{\omega \in \mathcal{R}} \sigma_{\min}(A - j\omega I). \quad (7.6.1)$$

So, for any real ω , $\sigma_{\min}(A - j\omega I)$ is an upper bound on $\beta(A)$; that is, $\beta(A) \leq \sigma_{\min}(A - j\omega I)$. Based on this idea, Van Loan (1985) gave two estimates for $\beta(A)$. One of them is a **heuristic estimate**:

$$\beta(A) \approx \min \{\sigma_{\min}(A - j\text{Re}(\lambda)I) \mid \lambda \in \Lambda(A)\}, \quad (7.6.2)$$

where $\Lambda(A)$ denotes the spectrum of A .

Thus, using this heuristic estimate, $\beta(A)$ may be estimated by finding the singular values of the matrix $(A - j\text{Re}(\lambda)I)$, for every eigenvalue λ of A . This approach was thought to give an upper bound within an order of magnitude of $\beta(A)$. However, Demmel (1987) has provided examples to show this bound can be larger than $\beta(A)$ by an arbitrary amount.

The other approach of Van Loan requires application of a general nonlinear minimization algorithm to $f(\omega) = \sigma_{\min}(A - j\omega I)$.

We will not pursue these approaches here. Rather, we will describe a simple bisection method to estimate $\beta(A)$ due to Byers (1988).

The bisection algorithm estimates $\beta(A)$ within a factor of 10 or indicates that $\beta(A)$ is less than a small tolerance. This is sufficient in practice. The algorithm makes use of the crude estimate of the upper bound $\beta(A) \leq \frac{1}{2}\|A + A^*\|_2$.

To describe the algorithm, let's define an $2n \times 2n$ **Hamiltonian matrix** $H(\sigma)$, given $\sigma \geq 0$, by

$$H(\sigma) = \begin{pmatrix} A & -\sigma I \\ \sigma I & -A^* \end{pmatrix}. \quad (7.6.3)$$

The bisection method is based on the following interesting spectral property of the matrix $H(\sigma)$.

For more on Hamiltonian matrices, see Chapter 10 and Chapter 13.

Theorem 7.6.1 $\sigma \geq \beta(A)$ if and only if $H(\sigma)$ defined by (7.6.3) has a purely imaginary eigenvalue.

Proof: Let ω_i be a purely imaginary eigenvalue of $H(\sigma)$. Then there exist nonzero complex vectors u, v such that

$$\begin{pmatrix} A & -\sigma I \\ \sigma I & -A^* \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \omega_i \begin{pmatrix} u \\ v \end{pmatrix}. \quad (7.6.4)$$

This gives us

$$(A - \omega_i I) u = \sigma v \quad (7.6.5)$$

and

$$(A - \omega_i I)^* v = \sigma u \quad (7.6.6)$$

This means that σ is a singular value of the matrix $A - \omega_i I$. Also, since $\beta(A) \leq \sigma_{min}(A - j\omega I)$ for any real ω , we obtain $\sigma \geq \beta(A)$.

Conversely, suppose that $\sigma \geq \beta(A)$. Define

$$f(\alpha) = \sigma_{min}(A - j\alpha I).$$

The function f is continuous and $\lim_{\alpha \rightarrow \infty} f(\alpha) = \infty$. Therefore, f has a minimum value $f(\alpha) = \beta(A) \leq \sigma$, for some real α .

By the Intermediate Value Theorem of Calculus, we then have $f(\omega) = \sigma$ for some real ω .

So, σ is a singular value of $A - j\omega I = A - \omega_i I$ and there exist unit complex vectors satisfying (7.6.5) and (7.6.6). This means that ω_i is a purely imaginary eigenvalue of $H(\sigma)$. ■

Algorithm 7.6.1 The Bisection Algorithm for Estimating the Distance to an Unstable System.

Inputs: A - An $n \times n$ stable complex matrix
 τ - Tolerance (> 0).

Outputs: Real numbers α and ν such that

either $\nu/10 \leq \alpha \leq \beta(A) \leq \nu$ or $0 = \alpha \leq \beta(A) \leq \nu \leq 10\tau$.

Step 1. Set $\alpha \equiv 0$, $\nu = \frac{1}{2}\|(A + A^*)\|_2$

Step 2. Do while $\nu > 10 \max(\tau, \alpha)$

$$\sigma \equiv \sqrt{\nu \max(\tau, \alpha)}$$

If $H(\sigma)$ has a purely imaginary eigenvalue, then set $\nu \equiv \sigma$; else $\alpha \equiv \sigma$

Example 7.6.1 Consider finding $\beta(A)$ for the matrix

$$A = \begin{pmatrix} -1 & 1 \\ 0 & -0.0001 \end{pmatrix}.$$

$$\tau = 0.0100$$

Iteration 1.

Step 1. Initialization: $\alpha = 0, \nu = 1.2071$.

Step 2. $10 \times \max(\tau, \alpha) = 0.0100$.

$$\sigma = 0.0059$$

$$H(\sigma) = \begin{pmatrix} -1 & 1 & -0.0059 & 0 \\ 0 & -0.0001 & 0 & -0.0059 \\ 0.0059 & 0 & 1 & 0 \\ 0 & 0.0059 & -1 & 0.0001 \end{pmatrix}$$

The eigenvalues of $H(\sigma)$ are: $-1, 1, \pm 0.0083j$. Since $H(\sigma)$ has an imaginary eigenvalue, we set $\nu = \sigma = 0.0059$.

Since $\nu = 0.0059$ is less than $10 \times \max(\tau, \alpha) = 0.0100$, we stop.

Conclusion: $\beta(A) \leq 0.0059 < 10\tau$.

Computational Remarks:

The bulk of the work of the Algorithm 7.6.1 is in deciding whether $H(\sigma)$ has an imaginary eigenvalue.

Also, the decision of whether $H(\sigma)$ has an imaginary eigenvalue in a computational setting (in the presence of round-off errors) is a tricky one. Some sort of threshold has to be used. However, if that decision is made in a numerically effective way, then in the worst case, “ $\beta(A)$ might lie outside the bound given by the algorithm by an amount proportional to the precision of the arithmetic” (Byers (1988)).

Because of the significant computational cost involved in deciding if the matrix $H(\sigma)$ at each step has an imaginary eigenvalue, **the algorithm may not be computationally feasible for large problems**.

Convergence. If $\tau = \frac{1}{2}10^{-p}\|A + A^*\|$, then at most $\log_2 p$ bisection steps are required; for example, if $\tau = \frac{1}{2} \times 10^{-8}\|A + A^*\|$, then at most three bisection steps are required. Algorithm 7.6.1 shows linear convergence, but can be modified to make it to a quadratically convergent algorithm. See Boyd and Balakrishnan (1990).

MATCONTROL NOTE. Algorithm 7.6.1 has been implemented in MATCONTROL function **disstabc**.

Relation to Lyapunov Equation

Since Lyapunov equations play a vital role in stability analysis of a linear system, it is natural to think that the distance to a set of unstable matrices $\beta(A)$ is also related to a solution of a Lyapunov equation. Indeed, the following result can be proved (See Malyshev and Sadkane (1999) and also Hewer and Kenney (1988)).

Theorem 7.6.2 (Distance to an Unstable System and Lyapunov Equation)

Let A be complex stable and let X be the unique positive Hermitian definite solution of the Lyapunov equation

$$XA + A^*X = -M, \quad (7.6.7)$$

where M is Hermitian positive definite. Then $\beta(A) \geq \frac{\lambda_{\min}(M)}{2\|X\|_2}$, where $\lambda_{\min}(M)$ denotes the smallest eigenvalue of M .

Proof: Let $\omega \in \mathbb{R}$ and u be a unit vector such that

$$\frac{1}{\beta(A)} = \max_{Re(z)=0} \| (A - zI)^{-1} \| = \| (A - j\omega I)^{-1} u \| \quad (7.6.8)$$

Let $x = (A - j\omega I)^{-1}u$. Then $\|x\|_2 = \frac{1}{\beta(A)}$.

Multiplying the Lyapunov equation (7.6.7) by x^* to the left and x to the right, we have

$$\begin{aligned} x^*(XA + A^*X)x &= -x^*Mx \\ x^*XAx + x^*A^*Xx &= -x^*Mx \end{aligned}$$

Then

$$x = (A - j\omega I)^{-1}u \Rightarrow (A - j\omega I)x = u \Rightarrow Ax = u + j\omega x$$

and

$$x = (A - j\omega I)^{-1}u \Rightarrow x^*(A - j\omega I)^* = u^* \Rightarrow x^*A^* + j\omega x^* = u^* \Rightarrow x^*A^* = u^* - j\omega x^*$$

Therefore

$$|x^*XAx + x^*A^*Xx| = |x^*X(u + j\omega x) + (u^* - j\omega x^*)Xx| = 2|u^*Xx| \leq 2\|X\|_2\|x\|_2. \quad (7.6.9a)$$

Also, by the Rayleigh quotient, we have, $\lambda_{\min}(M) \leq \frac{x^* M x}{x^* x}$, i.e.

$$\lambda_{\min}(M) \|x\|_2^2 \leq x^* M x = | - x^* M x|. \quad (7.6.9b)$$

Thus combining (7.6.9a) and (7.6.9b) yields

$$\lambda_{\min}(M) \|x\|_2^2 \leq |x^* M x| = |x^* (XA + A^* X)x| \leq 2\|X\|_2 \|x\|_2.$$

or

$$\lambda_{\min}(M) \|x\|_2 \leq 2\|X\|_2.$$

Since $\|x\|_2 = \frac{1}{\beta(A)}$, this means that $\lambda_{\min}(M) \frac{1}{\beta(A)} \leq 2\|X\|_2$ or $\beta(A) \geq \lambda_{\min}/2\|X\|_2$.

Example 7.6.2 Consider Example 7.6.1 again.

$$A = \begin{pmatrix} -1 & 1 \\ 0 & -0.0001 \end{pmatrix}.$$

Take $M = I_2$. Then $X = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 9999.5 \end{pmatrix}$.

By Theorem 7.6.2, $\beta(A) \geq 5.0002 \times 10^{-5}$.

Verify: The eigenvalues of $A + 5.0002 \times 10^{-5}I$ are -0.9999 and 0 .

Distance to a Discrete Unstable System

The discrete analog of $\beta(A)$ is defined to be

$$\gamma(A) = \min\{\|E\| \mid \text{for some } \theta \in \mathbb{R}; e^{i\theta} \in \Omega\Lambda(A + E)\}. \quad (7.6.9)$$

That is, **$\gamma(A)$ measures the distance from A to the nearest matrix with an eigenvalue on the unit circle**. If A is discrete-stable, then $\gamma(A)$ is a measure of how “nearly discrete-unstable” A is.

A discrete-analog of Theorem 7.6.1 is:

Theorem 7.6.3 Given an $n \times n$ complex matrix A , there exists a number $\Gamma(A) \in \mathbb{R}$ such that $\Gamma(A) \geq \gamma(A)$ and for $\Gamma(A) \geq \sigma \geq \gamma(A)$, the $2n \times 2n$ Hamiltonian matrix pencil

$$H_D(\sigma) = F(\sigma) - \lambda G(\sigma) = \begin{pmatrix} -\sigma I_n & A \\ I_n & 0 \end{pmatrix} - \lambda \begin{pmatrix} 0 & I_n \\ A^T & -\sigma I_n \end{pmatrix}$$

has a generalized eigenvalue of magnitude 1. Furthermore, if $\sigma < \gamma(A)$, then the above pencil has no generalized eigenvalue of magnitude 1.

Proof: See Byers (1988).

Based on the above result, Byers (1988) described the following bisection algorithm to compute $\gamma(A)$, analogous to Algorithm 7.6.1.

The algorithm estimates $\gamma(A)$ within a factor of 10 or indicates that $\gamma(A)$ is less than a tolerance. The algorithm uses a crude bound $\Gamma(A) \geq \sigma_{\min}(A - I)$.

Algorithm 7.6.2 *The Bisection Algorithm for Estimating the Distance to a Discrete Unstable System.*

Inputs: An $n \times n$ complex matrix A and a tolerance $\tau > 0$.

Outputs. Real numbers α and δ such that $\frac{\delta}{10} \leq \alpha \leq \gamma(A) \leq \delta$ or $0 = \alpha \leq \gamma(A) \leq \delta \leq 10\tau$.

Step 1. Set $\alpha \equiv 0$; $\delta \equiv \sigma_{\min}(A - I)$

Step 2. Do while $\delta > 10 \max(\tau, \alpha)$

$$\sigma \equiv \sqrt{\delta \max(\tau, \alpha)}.$$

If the pencil $F(\sigma) - \lambda G(\sigma)$, defined above, has a generalized eigenvalue of magnitude 1, then set $\delta \equiv \sigma$, else $\alpha \equiv \sigma$.

End.

Example 7.6.3 Let $A = \begin{pmatrix} 0.9999 & 1 \\ 0 & 0.5 \end{pmatrix}$, $\tau = 10^{-8}$. The matrix A is discrete-stable.

Iteration 1:

Step 1: $\alpha = 0$, $\delta = 4.4721 \times 10^{-5}$

Step 2: $\delta > 10 \max(\tau, \alpha)$ is verified, we compute $\sigma = 6.6874 \times 10^{-6}$. The eigenvalues of $H_D(\sigma)$ are 2, 1.0001, 0.9999 and 0.5000. Thus, $\alpha \equiv 6.6874 \times 10^{-6}$.

Iteration 2: Since $\delta > 10 \max(\tau, \alpha)$ is verified, we compute $\sigma = 5.4687 \times 10^{-6}$. The eigenvalues of $H_D(\sigma)$ are 2, 1.001, 0.999, 0.5000; we set $\alpha = \sigma = 5.4687 \times 10^{-6}$.

Iteration 3: $\delta < 10 \max(\tau, \alpha)$, the iteration stops, and on exit we obtain

$$\alpha = 5.4687 \times 10^{-6}$$

$$\delta = 4.4721 \times 10^{-5}.$$

MATCONTROL Note: Algorithm 7.6.2 has been implemented in MATCONTROL function **disstabd**.

7.7 Robust Stability

Even though a system is known to be stable, it is important to investigate if the system remains stable under certain perturbations. Note that in most physical systems, the system matrix A

is not known exactly; what is known is $A + E$, where E is an $n \times n$ perturbation matrix. Thus, in this case the stability problem becomes the problem of finding the system

$$\dot{x}(t) = (A + E)x(t) \quad (7.7.1)$$

remains stable, given that A is stable.

The solution of the Lyapunov equations again can be used to obtain bounds on the perturbation that guarantee that the perturbed system (7.7.1) remains stable.

The first of such results was obtained by Patel and Toda (1980).

In the following theorems $\sigma_{\max}(M)$, as usual, stands for the largest singular value of M .

Theorem 7.7.1 *Let A be a stable matrix. Thus $(A + E)$ remains stable if*

$$\sigma_{\max}(E) < \frac{1}{\sigma_{\max}(X)},$$

where X is a unique symmetric positive definite solution of the Lyapunov equation

$$XA + A^T X = -2I. \quad (7.7.2)$$

■

The bound of Theorem 7.7.1 was improved by Yedavalli (1985), who proved the following result:

Theorem 7.7.2 *Let A be a stable matrix. Let E_A be an $n \times n$ matrix formed out of the perturbation matrix E such that the (i, j) -th entry of E_A is zero if $e_{ij} = 0$, and is 1 if $e_{ij} \neq 0$. Let X be the symmetric positive definite solution of the Lyapunov equation (7.7.2):*

Then $A + E$ is stable if

$$|e_{ij}|_{\max} = \delta < \frac{1}{\sigma_{\max}(Y)}$$

where $Y = \frac{1}{2}(Z + Z^T)$ and $Z = |X|E_A$.

Here $|X|$ is the matrix obtained from X by taking the absolute values of the entries of X . ■

Based on the above theorem, we can state the following algorithm for robust stability analysis.

Algorithm 7.7.1 Robust Stability Analysis Using Lyapunov Equation

Input: A -The stable state matrix.

$E = (e_{ij})$ – The perturbation matrix

Output: Decision on the stability of $A + E$.

Step 1. Form the auxiliary error matrix E_A such that the (i, j) th entry of E_A is zero if $e_{ij} = 0$, is 1 if $e_{ij} \neq 0$.

Step 2. Solve the Lyapunov equation

$$XA + A^T X = -2I.$$

Step 3. Form $Z = |X|E_A$ and $Y = \frac{1}{2}(Z + Z^T)$.

Step 4. Compute $\delta' = \frac{1}{\sigma_{\max}(Y)}$.

Step 5. Check if $\delta = |e_{ij}|_{\max}$ is less than δ'

Step 6. Decision. If $\delta < \delta'$, then $A + E$ is stable.

Example 7.7.1

$$A = \begin{pmatrix} -4.1793 & 9.7472 & 1.3649 \\ 0 & -1.0827 & 0.3796 \\ 0 & 0 & -9.4673 \end{pmatrix}, \quad E = \begin{pmatrix} 0.0668 & 0.0120 & 0.0262 \\ 0.0935 & 0.0202 & 0.0298 \\ 0.0412 & 0.0103 & 0.0313 \end{pmatrix}$$

Step 1

Form the auxiliary error matrix

$$E_A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Step 2

Solve the Lyapunov equation

$$A^T X + X A = -2I$$

$$X = \begin{pmatrix} 0.2393 & 0.4432 & 0.0363 \\ 0.4432 & 4.9139 & 0.2676 \\ 0.0363 & 0.2676 & 0.1216 \end{pmatrix}.$$

Step 3.

compute $Z = |X|E_A$.

$$Z = \begin{pmatrix} 0.7188 & 0.7188 & 0.7188 \\ 5.6248 & 5.6248 & 5.6248 \\ 0.4255 & 0.4255 & 0.4255 \end{pmatrix}.$$

Compute $Y = \frac{Z + Z^T}{2}$.

$$Y = \begin{pmatrix} 0.7188 & 0.1718 & 3.5721 \\ 3.1718 & 5.6248 & 3.0251 \\ 0.5721 & 3.0251 & 0.4255 \end{pmatrix}.$$

Step 4

Compute $\delta = |E_{ij}|_{\max} = 0.0935$

Compute $\delta' = \frac{1}{\sigma_{\max}(Y)} = 0.1203$

Step 5 and Step 6.

Since $\delta = 0.0935 < 0.1203 = \delta'$, $A + E$ is stable.

Note: The eigenvalues of $A + E$ are $-4.3856, -0.7817, -9.4437$. The Lyapunov equation in Step 2 was solved using the MATLAB function `lyap`.

MATCONTROL Note: Algorithm 7.7.1 has been implemented in MATCONTROL function `robstab`.

We next state a more general result on robust stability due to Keel, Bhattacharyya, and Howze (1988). The proof can be found in Bhattacharyya et al. (1995, 519-520). The result there is proved in the context of feedback stabilization, and we will revisit the result later in that context.

Theorem 7.7.3 Let A be a stable matrix and let the perturbation matrix E be given by

$$E = \sum_{i=1}^r p_i E_i, \quad (7.7.3)$$

where $E_i, i = 1, \dots, r$ are matrices determined by structure of the perturbations.

Let Q be a symmetric positive definite matrix and X be a unique symmetric positive definite solution of the Lyapunov equation

$$XA + A^T X + Q = 0.$$

Then the system (7.7.1) remains asymptotically stable for all p_i satisfying

$$\sum_{i=1}^r |p_i|^2 < \frac{\sigma_{\min}^2(Q)}{\sum_{i=1}^r \mu_i^2}, \quad (7.7.4)$$

where $\sigma_{\min}(Q)$ denotes the minimum singular value of Q and μ_i is given by

$$\mu_i = \|E_i^T X + X E_i\|_2. \blacksquare$$

Example 7.7.2 Let $r = 1, p_1 = 1$; so that $E = E_1$.

Choose the matrix A and the matrix E the same as in Example 7.7.1.

Choose $Q = 2I$ so that the Lyapunov equation to be solved is also the same as in Example 7.7.1.

Then $\mu_1 = \|E_1^T X + X E_1\|_2 = 0.7199$, and the right hand side of (7.7.4) is 7.7185.

Since $|p_1^2| = 1 < 7.7185$, the matrix $A + E$ is stable.

Remarks: For the above values of r and p_1 , we have thus recovered the same result as we obtained from Theorem 7.7.2.

However, note that Theorem 7.7.3 allows us to have much larger perturbations to A while maintaining stability.

For example if we choose, $r = 1, p_1 = 2$, then since $|p_1|^2 = 4 < 7.7184$, we have by Theorem 7.7.3 that $A + 2E$ is also stable.

A result similar to that stated in Theorem 7.7.3, was also proved by Zhou and Khargonekar (1987). We state the result below. For the proof of the result, we refer the readers to their paper.

Theorem 7.7.4 *Let A be a stable matrix and let E be given by (7.7.3). Let X be the unique symmetric positive definite solution of (7.7.2). Define*

$$X_i = (E_i^T X + X E_i)/2, \quad i = 1, 2, \dots, r \quad (7.7.5)$$

and

$$X_e = (X_1, X_2, \dots, X_r).$$

Then (7.7.1) remains stable if

$$\sum_{k=1}^r p_k^2 < \frac{1}{\sigma_{max}^2(X_e)} \text{ or } \sum_{i=1}^r |p_i| \sigma_{max}(X_i) < 1 \text{ or } |p_i| < \frac{1}{\sigma_{max} \left(\sum_{i=1}^r |X_i| \right)}, \quad i = 1, \dots, r. \quad (7.7.6)$$

Remark: It should be noted that Theorems 7.7.1-7.7.4 all give only **sufficient conditions** for robust stability. Algorithm 7.7.1 is thus only a sufficient test. A number of other sufficient conditions can be found in the book by Boyd, El Ghaoui, Feron and Balakrishnan (1994).

Robust Stability of Polynomials

Robust stability of polynomials has been the subject of extensive research in recent years. A remarkable result due to a Russian Mathematician Kharitonov has been the focal point of this research.

Since in this book we are mainly concerned with the stability analysis of a matrix and the polynomial approach is not discussed, we will not discuss the robust stability of polynomials in details.

We will just state below the original result of Kharitonov (1978) without proof. For details and more up-to-date results, we refer the readers to the recent books on the subject: Barmish (1994), Bhattacharyya et al. (1995), Gutman (1990), etc. In particular, a proof of Theorem 7.7.5 can be found in Bhattacharyya et al. (1995, pp. 224-230).

Theorem 7.7.5 (The Kharitonov Theorem on Robust Stability). Let $\mathcal{P}(s)$ be the set of real polynomials of degree n of the form

$$p(s) = p_0 + p_1 s + p_2 s^2 + \cdots + p_n s^n$$

where the coefficients lie within the ranges

$$p_i(s) \in [x_i, y_i], i = 0, 1, \dots, n.$$

Then every polynomial in the set $\mathcal{P}(s)$ is Hurwitz (that is, all the zeros have negative real parts) if and only if the following four extreme polynomials are Hurwitz:

$$\begin{aligned} p^{(1)}(s) &= x_0 + x_1 s + y_2 s^2 + y_3 s^3 + x_4 s^4 + x_5 s^5 + y_6 s^6 + \cdots \\ p^{(2)}(s) &= x_0 + y_1 s + y_2 s^2 + x_3 s^3 + x_4 s^4 + y_5 s^5 + y_6 s^6 + \cdots \\ p^{(3)}(s) &= y_0 + x_1 s + x_2 s^2 + y_3 s^3 + y_4 s^4 + x_5 s^5 + x_6 s^6 + \cdots \\ p^{(4)}(s) &= y_0 + y_1 s + x_2 s^2 + x_3 s^3 + y_4 s^4 + y_5 s^5 + x_6 s^6 + \cdots \end{aligned}$$

The polynomials $p^{(i)}(s)$, $i = 1, 2, 3, 4$ are called **Kharitonov polynomials**.

Example 7.7.3 Consider

$$p(s) = p_0 + p_1 s + p_2 s^2 + p_3 s^3 + p_4 s^4 + p_5 s^5$$

and

$$p_0 \in [0.25, 1.25], \quad p_1 \in [0.75, 1.25],$$

$$p_2 \in [2.75, 3.25], \quad p_3 \in [0.25, 1.25].$$

The four Kharitonov polynomials are:

$$\begin{aligned} p^{(1)}(s) &= 1.25s^3 + 3.25s^2 + 0.75s + 0.25 \\ p^{(2)}(s) &= 0.25s^3 + 3.25s^2 + 1.25s + 0.25 \\ p^{(3)}(s) &= 1.25s^3 + 2.75s^2 + 0.75s + 1.25 \\ p^{(4)}(s) &= 0.25s^3 + 2.75s^2 + 1.25s + 1.25 \end{aligned}$$

Using the Routh-Hurwitz or any other stability criterion of a single polynomial, it is easy to check that all these four Kharitonov polynomials are Hurwitz (stable). Thus every member of the family $p(s)$ is stable.

7.8 The Structured Stability Radius

In section 7.6 we introduced the concept of the distance of a stable matrix from the set of unstable matrices.

Here we specialize this concept to “**structured stability**,” meaning that we are now interested in finding the distance from a stable matrix to the set of unstable matrices, **where the distance**

is measured by the size of the additive perturbations of the form $B\Delta C$, with B and C fixed, and Δ variable.

Let A, B , and C be, respectively, $n \times n, n \times m$ and $r \times n$ matrices over the field \mathbb{F} (\mathbb{F} can be \mathbb{C} or \mathbb{R}). Then the (structured) **stability radius** of the matrix triple (A, B, C) is defined as

$$r_{\mathbb{F}}(A, B, C) = \inf\{\bar{\sigma}(\Delta) : \Delta \in \mathbb{F}^{m \times r} \text{ and } A + B\Delta C \text{ is unstable}\}, \quad (7.8.1)$$

where $\bar{\sigma}(M)$ following the notation of Qiu et al (1995), denotes the largest singular value of M (that is, $\bar{\sigma}(M) = \sigma_{max}(M)$). For real matrices (A, B, C) , $r_{\mathbb{R}}(A, B, C)$ is called the **real stability radius** and, for complex matrices (A, B, C) , $r_{\mathbb{C}}(A, B, C)$ is called the **complex stability radius**.

The stability radius, thus, determines the magnitude of the smallest perturbation needed to destroy the stability of the system.

“Stability” here is referred to as either continuous-stability (with respect to the left half-plane) or discrete-stability (with respect to the unit circle).

Let $\partial\mathbb{C}_g$ denote the boundary of either the half plane or the unit circle. Let A be stable or discrete-stable.

Then

$$\begin{aligned} r_{\mathbb{F}}(A, B, C) &= \inf\{\bar{\sigma}(\Delta) | \Delta \in \mathbb{F}^{m \times r} \text{ and } A + B\Delta C \text{ has an eigenvalue on } \partial\mathbb{C}_g\}. \\ &= \inf_{s \in \partial\mathbb{C}_g} \inf\{\bar{\sigma}(\Delta) | \Delta \in \mathbb{F}^{m \times r} \text{ and } \det(sI - A - B\Delta C) = 0\} \\ &= \inf_{s \in \partial\mathbb{C}_g} \inf\{\bar{\sigma}(\Delta) | \Delta \in \mathbb{F}^{m \times r} \text{ and } \det(I - \Delta G(s)) = 0\}, \end{aligned} \quad (7.8.2)$$

where $G(s) = C(sI - A)^{-1}B$.

Thus, given a complex $r \times m$ matrix M , the stability radius problem reduces to the problem of computing

$$\mu_{\mathbb{F}}(M) = [\inf\{\bar{\sigma}(\Delta) : \Delta \in \mathbb{F}^{m \times r} \text{ and } \det(I - \Delta M) = 0\}]^{-1}.$$

The Complex Stability Radius

It is easy to see that

$$\mu_{\mathbb{C}}(M) = \bar{\sigma}(M).$$

Thus we have the following formula for the complex stability radius.

Theorem 7.8.1 (The Complex Stability Radius Formula)

$$r_{\mathbb{C}}(A, B, C) = \left\{ \sup_{s \in \partial\mathbb{C}_g} \bar{\sigma}(G(s)) \right\}^{-1}. \blacksquare \quad (7.8.3)$$

The Real Stability Radius

If \mathbb{F} is \mathbb{R} , then according to above we have

$$r_{\mathbb{R}}(A, B, C) = \left\{ \sup_{s \in \partial \mathbb{C}_g} \mu_{\mathbb{R}}[C(sI - A)^{-1}B] \right\}^{-1}. \blacksquare \quad (7.8.4)$$

The following important formula for computing $\mu_{\mathbb{R}}(M)$ has been recently obtained by Qiu, Bernhardsson, Rantzer, Davison, Young, and Doyle (1995). We quote the formula from this paper. The proof is involved and we refer the readers to the paper for the proof. Following the notation of this paper, **we denote the second largest singular value of M by $\sigma_2(M)$** , and so on.

Denote the real and imaginary parts of a complex matrix M by $Re(M)$ and $Im(M)$, respectively. That is, $M = Re(M) + jIm(M)$.

Then the following result holds:

$$\mu_{\mathbb{R}}(M) = \inf_{\gamma \in (0, 1]} \sigma_2 \left(\begin{bmatrix} Re(M) & -\gamma Im(M) \\ \frac{1}{\gamma} Im(M) & Re(M) \end{bmatrix} \right) \quad (7.8.5)$$

The function to be minimized is a unimodular function on $(0, 1]$.

Furthermore, if $Im(M) = I$, then

$$\mu_{\mathbb{R}}(M) = \max\{\bar{\sigma}(U_2^T Re(M)), \bar{\sigma}(Re(M)V_2)\},$$

where U_2 and V_2 are defined by the singular value decomposition of $Im(M)$; that is, they satisfy

$$Im(M) = [U_1, U_2] \begin{bmatrix} \bar{\sigma}(Im(M)) & 0 \\ 0 & 0 \end{bmatrix} [V_1, V_2]^T.$$

Thus we have the following formula for the real stability radius.

Theorem 7.8.2 (The Real Stability Radius Formula)

$$r_{\mathbb{R}}(A, B, C) = \inf_{s \in \partial \mathbb{C}_g} \inf_{\gamma \in (0, 1]} \sigma_2 \left(\begin{bmatrix} Re(G(s)) & -\gamma Im(G(s)) \\ \frac{1}{\gamma} Im(G(s)) & Re(G(s)) \end{bmatrix} \right), \quad (7.8.6)$$

where $G(s) = C(sI - A)^{-1}B$. ■

Note that since the function to be minimized is unimodular, any local minimum is also a global minimum.

Notes:

(i)

$$r_{\mathbb{R}}(A, B, C) \geq r_{\mathbb{C}}(A, B, C). \quad (7.8.7)$$

(ii) The ratio $\frac{r_{\mathbb{R}}(A, B, C)}{r_{\mathbb{C}}(A, B, C)}$ can be arbitrarily large.

The following example taken from Hinrichsen and Pritchard (1990) illustrates (ii).

Example 7.8.1 Let

$$A = \begin{pmatrix} 0 & 1 \\ -1 & -\epsilon \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ -\epsilon \end{pmatrix},$$

and

$$C = (1, 0).$$

Then the transfer function

$$\begin{aligned} G(s) &= C(j\omega I - A)^{-1}B \\ &= \frac{-\epsilon}{1 - \omega^2 + j\omega\epsilon}. \end{aligned}$$

By (7.8.4) the real stability radius

$$r_{\mathbb{R}}(A, B, C) = \frac{1}{\epsilon}.$$

Since $|G(j\omega)|^2 = \frac{\epsilon^2}{(1 - \omega^2)^2 + \epsilon^2\omega^2}$, it is easy to see that $|G(j\omega)|^2$ is maximized when $\omega^2 = 1 - \frac{\epsilon^2}{2}$, if $\epsilon < \sqrt{2}$. So, by (7.8.3)

$$r_{\mathbb{C}}^2(A, B, C) = 1 - \frac{\epsilon^2}{4}.$$

Thus, if ϵ is considered as a parameter, then $r_{\mathbb{C}}(A, B, C)$ is always bounded by 1 whereas $r_{\mathbb{R}}(A, B, C)$ can be made arbitrarily large by choosing ϵ small enough.

Specialization of the Stability Radius to the Distance from Unstable Matrices

From (7.8.3) we immediately have the following relation between the distance to an unstable system and the stability radius:

$$\beta = r_{\mathbb{C}}(A, I, I) = \min_{\omega \in \mathbb{R}} \sigma_{\min}(A - j\omega I) = \beta(A).$$

Also, using Theorem 7.8.2, the following formula for $\beta(A)$, when A is a real stable matrix, can be proved [Exercise 23].

Theorem 7.8.3 Let A be a real stable matrix. Then

$$\beta(A) \equiv r_{\mathbb{R}}(A, I, I) = \min_{s \in \partial \mathbb{C}_g} \max_{\gamma \in (0, 1]} \sigma_{2n-1} \left(\begin{bmatrix} A - Re(sI) & -\gamma Im(sI) \\ -\frac{1}{\gamma} Im(sI) & A - Re(sI) \end{bmatrix} \right). \quad (7.8.8)$$

■

Note: For each fixed s the function in (7.8.8) to be maximized is quasiconcave.

7.9 Some Selected Software

7.9.1 MATLAB CONTROL SYSTEM TOOLBOX

- | | |
|---------|---|
| norm | - Computes the H_2 -norm of the system. |
| bode | - Computes the magnitude and phase of the frequency response, which are used to analyze stability and robust stability. |
| nyquist | - Calculates the Nyquist frequency response. System properties such as gain margin , phase margin , and stability can be analyzed using Nyquist plots. (The gain margin and phase margin are widely used in classical control theory as measures of robust stability). |

7.9.2 MATCONTROL

- | | |
|----------|--|
| INERTIA | - Determining the inertia and stability of a matrix without solving a matrix equation or computing eigenvalues |
| H2NRMCG | - Finding H_2 -norm using the controllability Grammians |
| H2NRMOG | - Finding H_2 -norm using the observability Grammian |
| DISSTABC | - Determining the distance to the continuous-time stability |
| DISSTABD | - Determining the distance to the discrete-time stability |
| ROBstab | - Robust stability analysis using Lyapunov equations |

7.9.3 SLICOT

- | | |
|--------|--|
| AB13BD | - H_2 or L_2 norm of a system. |
| AB13ED | - Complex Stability radius, using bisection. |
| AB13FD | - Complex Stability radius, using bisection and SVD. |

7.10 Summary and Review

The stability of the system

$$\dot{x}(t) = Ax(t)$$

or that of

$$x(k+1) = Ax(k)$$

is essentially governed by the eigenvalues of the matrix A .

I. Mathematical Criteria of Stability

The continuous-time system $\dot{x}(t) = Ax(t)$ is asymptotically stable if and only if the eigenvalues of A are all in the left half plane (**Theorem 7.2.1**).

Similarly, the discrete-time system $x(k+1) = Ax(k)$ is asymptotically stable if and only if all the eigenvalues of A are inside the unit circle. (**Theorem 7.3.1**).

Various Lyapunov stability theorems (**Theorems 7.2.3-7.2.9**, and **Theorem 7.3.2**) have been stated and proved.

II. The inertia of a Matrix

Several important inertia theorems (**Theorems 7.4.1 - 7.4.6**) including the classical **Sylvester Law of Inertia** and those that generalize the Lyapunov stability theorems have been stated and some have been proved.

III. Methods for Determining Stability and Inertia

The **Characteristic Polynomial Approach** and the **Matrix Equation Approach** are two classical approaches for determining the stability of a system and the inertia of a matrix. Both these approaches have some computational drawbacks.

The zeros of a polynomial may be extremely sensitive to small perturbations. Furthermore, the numerical methods to compute the characteristic polynomial of a matrix are usually unstable.

The most numerically effective method (**The Schur method**, described in Chapter 8), for solving a Lyapunov matrix equation is based on reduction of the matrix A to real Schur form, and the real Schur form displays the eigenvalues of A or the eigenvalues can be trivially computed out of this form.

Thus the characteristic equation approach is not numerically viable and the matrix equation approach for stability and inertia is counterproductive.

Hence, the most numerically effective approach for stability and inertia is the eigenvalue approach: compute all the eigenvalues of A .

By explicitly computing the eigenvalues, one, however, gets much more than what is needed for stability and inertia. Furthermore, since the eigenvalues of a matrix can be very sensitive to small perturbations, determining the inertia and stability by computing explicitly the eigenvalues can be misleading.

An implicit matrix equation approach (**Algorithm 7.5.1**), which does not require computation of eigenvalues nor explicit solution of any matrix equation has been described. **Algorithm 7.5.1 is about three times faster than the eigenvalue method.**

IV. Distance to an Unstable System

Given a stable matrix A , the quantity $\beta(A)$ defined by

$$\beta(A) = \min \{ \|E\|_F \text{ such that } A + E \in U \},$$

where U is the set of $n \times n$ matrices with at least one eigenvalue on the imaginary axis, is the distance to the set of unstable matrices.

A bisection algorithm (**Algorithm 7.6.1**), based on knowing if a certain Hamiltonian matrix (the matrix (7.6.3)) has a purely imaginary eigenvalue, is described. The algorithm

is based on **Theorem 7.6.1**, which displays a relationship between a spectral property of the Hamiltonian matrix and the quantity $\beta(A)$.

The discrete-analog of $\beta(A)$ is defined to be

$$\gamma(A) = \min\{||E|| \text{ for some } \theta \in \mathbb{R}; e^{i\theta} \in \Omega(A + E)\}.$$

An analog of Theorem 7.6.1 (**Theorem 7.6.3**) is stated and a bisection algorithm (**Algorithm 7.6.2**) based on this theorem is described.

V. Robust Stability

Given a stable matrix A , one naturally wonders if the matrix $A + E$ remains stable, where E is a certain perturbed matrix. Several bounds for E guaranteeing the stability of the perturbed matrix $(A + E)$ are given, in terms of solutions of certain Lyapunov equations (**Theorems 7.7.1-7.7.4**).

Also, the well-known **Kharitonov theorem** (**Theorem 7.7.5**) on robust stability of a polynomial is stated without a proof. This remarkable theorem states that the stability of a family of polynomials whose coefficients lie in an interval is determined by the stability of only four extreme polynomials.

VI. Stability Radius

Section 7.8 deals with the **structured stability radius**. If the perturbations are of the form $B\Delta C$, where Δ is an unknown perturbation matrix, then it is of interest to know the size of smallest Δ (measured using 2-norm) that will destabilize the perturbed matrix $A + B\Delta C$. In this context, the concept of **stability radius** is introduced, and formulas both for the complex stability radius (**Theorem 7.8.1**) and the real stability radius (**Theorem 7.8.2**) are stated.

VII. H_2 -Norm. The H_2 -Norm of a stable transfer, transfer function measures the steady-state covariance of the output response $y = Gv$ to the white noise inputs v . An algorithm for computing the H_2 -norm, based on computing the controllability or observability Grammian via Lyapunov equations is given.

7.11 Chapter Notes and Further Reading

A voluminous work has been published on Lyapunov stability theory since the historical monograph “**Problème de la stabilité du Mouvement**” was published by the Russian mathematician A. M. Liapunov in 1892. Some of the books that exclusively deal with Lyapunov stability are those by LaSalle and Lefschetz (1961), Lehnigk (1966), etc., and a good account of diagonal stability and diagonal-type Lyapunov functions appears in the recent book by Kaszkurewicz and Bhaya (1999). For a good account of BIBO and BIBS stability, see the book by DeCarlo (1989).

In Section 7.2, we have just given a very brief account of the Lyapunov stability adapted to the linear case. The matrix equation version in the linear case seems to have first appeared in the book by Gantmacher (1959, Vol. II). There exists many proofs of Lyapunov stability Theorem (**Theorem 7.2.3**). The proof given here is along the line in Bellman (1960).

The proofs of the other theorems in this section can be found in most linear systems books, including the books by Chen (1984), Kailath (1980), Wonham (1986), etc.

The inertia theory has been mainly confined to the linear algebra literature. An excellent account of its control theoretic applications appear in Glover (1984) and in the book by Zhou et al. (1996).

There are also a few papers on the inertia theory with respect to more general regions in the complex plane other than the half-planes and the unit circle given in Section 7.4. Inertia theory has been applied to obtain elementary proofs of several classical root-location problems in Datta (1978a, 1978b, 1979). For an account of this work, see the recent survey paper of the author (Datta 1999). The inertia and stability algorithm is due to Carlson and Datta (1979). The algorithm has been modified by Datta and Datta (1987) and extended to other regions in the complex plane in Datta and Datta (1986).

The concept of distance to instability was perhaps introduced by Van Loan (1985). The bisection algorithm (**Algorithm 7.6.1**) is due to Byers (1988).

There are now several good books on robust control. These include the books by Bhattacharyya et al. (1995), Green and Limebeer (1995), Zhou et al (1996), Barmish (1994), Hinrichsen and Martensson (1990). Dorato and Yedavalli (1989). The concept of complex stability radius as robustness measures for stable matrices (in the form given here) was introduced by Hinrichsen and Pritchard (1986). There are several good papers on this subject in the book “**Control of uncertain systems**”, edited by Hinrichsen and Martensson (1990). Discussion of Section 7.8 has been taken from Qiu et al. (1995).

EXERCISES

1. Verify that the spring-mass system of Example 5.2.2 is not asymptotically stable;
What is the physical interpretation of the above statement?
2. Consider the problem of cart with two sticks considered in Exercise 3 of Chapter 5. Take $M_1 = M_2 = M$.
 - (a) Show that at the equilibrium states, \bar{x}_1 and \bar{x}_2 are nonzero, and $\bar{x}_3 = \bar{x}_4 = 0$. What is the physical significance of this?
 - (b) Show that the system is not asymptotically stable.
3. Consider the stick-balancing problem in Example 5.2.4. Give a mathematical explanation of the fact that without an input to the control system, if the stick is not upright with zero velocity, it will fall.
4. For the Cohort Population Model of Example 5.2.7, derive a set of sufficient conditions, each of which guarantees the asymptotic stability of the system.
5. Give a proof of Theorem 7.3.2 from that of Theorem 7.2.3 using the matrix version of the Cayley transformation.
6. Prove that the system (7.2.2) is BIBO if and only if $G(s) = C(sI - A)^{-1}B$ has every pole with negative real part.
7. Prove that the discrete-time system

$$x_{k+1} = Ax_k + Bu_k$$

is BIBO stable if and only if all the poles of the transfer functions lie inside the open unit circle of the z -plane.

8. Prove that the discrete-time system in Exercise 7 is BIBS if and only if (i) all the eigenvalues of A lie in the closed unit disc, (ii) the eigenvalues on the unit disc have multiplicity 1 in the minimal polynomial of A , and (iii) the unit circle modes are uncontrollable (consult DeCarlo (1989), pp.422).
9. Let X and M be the symmetric positive definite matrices such that

$$XA + A^T X + 2\lambda X = -M,$$

then prove that all eigenvalues of A have a real part that is less than $-\lambda$.

10. Prove that A is a stable matrix if and only if $\|e^{At}\| \leq k$ for some $k > 0$.

11. Prove that if M is positive definite and the discrete Lyapunov equation

$$X - A^T X A = M$$

has a symmetric positive definite solution X , then A is discrete-stable.

12. Prove the following results:

- (a) Suppose that A is discrete-stable. Then (A, B) is controllable if and only if the discrete Lyapunov equation

$$X - AXA^T = BB^T$$

has a unique positive definite solution.

- (b) Suppose that (A, B) is controllable. Then A is discrete-stable if and only if the discrete Lyapunov equation

$$X - AXA^T = BB^T$$

has a unique positive definite solution

- (c) Suppose that (A, C) is observable. Then A is discrete-stable if and only if there exists a unique positive definite solution X of the discrete Lyapunov equation

$$X - A^T X A = C^T C.$$

13. (Glover (1984)).

Let $X = X^T = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix}$ with $\delta(X) = 0$. Suppose that

$$\begin{aligned} AX + XA^T &= -BB^T \\ A^T X + XA &= -C^T C. \end{aligned}$$

Partition

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix},$$

and $C = (C_1, C_2)$, conformably with X . Then prove the following:

- (1) If $\gamma(X_1) = 1$, then $\pi(A_{11}) = 0$
- (2) If $\delta(A) = 0$ and $\lambda_i(X_1^2) \neq \lambda_j(X_2^2) \forall i, j$, then $In(A_{11}) = In(X_1)$ and $In(A_{22}) = In(-X_2)$. (Here $\lambda_i(M)$ denotes the i th eigenvalue of M).

14. Prove that in Theorem 7.4.5, the assumption that (A^T, M) is controllable implies that $\delta(X) = 0$.

15. Let A be a stable matrix. Prove that (i) $\beta(A) = \min_{\omega \in R} \sigma_{\min}(A - j\omega I)$, (ii) $\beta(A) \leq \sigma_{\min}(A)$, (iii) $\beta(A) \leq |\alpha(A)|$, where $\alpha(A) = \max\{Re(\lambda) | \lambda \text{ is an eigenvalue of } A\}$.
16. Give an example to show that the bound of $\beta(A)$ given by (7.6.2) can be arbitrary large (Consult the paper of Demmel (1987)).
17. Construct an example to show that a matrix A can be very near to an unstable matrix without $\alpha(A)$, defined in Exercise 12, being small.
18. Let $Arg(z)$ represent the argument of the complex number z . Let $r > 0$ and $\rho \in \mathbb{C}$, then prove that $r^{-1}\gamma(r(A + \rho I))$ is the distance from A to the nearest matrix with an eigenvalue on the circle $\{z \in C/|z - \rho| = r^{-1}\}$, where $\gamma(M)$ denotes the distance of a discrete-stable matrix M to instability, defined by (7.6.10).

Use the result to develop an algorithm to estimate this quantity.

19. Give proofs of Theorems 7.7.1-7.7.4 (consult the associated papers, as necessary).
20. Consider the perturbed system:

$$\dot{x} = (A + BKC)x,$$

where $A = \text{diag}(-1, -2, -3)$,

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$

$K = \begin{pmatrix} -1 + k_1 & 0 \\ 0 & -1 + k_2 \end{pmatrix}$, and k_1 and k_2 are two uncertain parameters varying in the intervals around zero. Use each of the Theorems 7.7.1-7.7.4 to calculate and compare the allowable bounds on k_1 and k_2 that guarantee the stability of $A + BKC$.

21. Construct an example to verify each of the followings:
- (i) The real stability radius is always greater than or equal to the complex stability radius.
 - (ii) The ratio of the real stability radius to the complex stability radius can be made arbitrarily large.

22. Prove that the H_2 -norm of the discrete-time transfer matrix $G(z) = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix}$ can be computed as

$$\|G(z)\|_2^2 = \text{Trace}(CC_G^D C^T) = B^T O_G^D B$$

where C_G^D and O_G^D are, respectively, the discrete-time controllability and observability Grammians, given by (7.3.6) and (7.3.7), respectively. Write down a Lyapunov equation

based algorithm to compute the H_2 -norm of a discrete-time system based on the above formula.

23. Give a proof of Theorem 7.8.3.

References

1. J. Ackermann, *Robust Control: Systems with Uncertain Physical Parameters*, Springer-Verlag, New York, NY, 1993.
2. B.R. Barmish, *New Tools for Robustness of Linear Systems*, McMillan Publishing Co., New York, NY 1994.
3. R. Bellman, *Introduction to Matrix Analysis*, McGraw Hill, New York, 1960.
4. S.P. Bhattacharyya, H. Chapellat and L.H. Keel, *Robust Control: the Parametric Approach*, Prentice Hall Information and Systems Sciences Series, Prentice Hall, Upper Saddle River, NJ, 1995, (Thomas Kailath, Editor).
5. S. Boyd and V. Balakrishnan, A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithms for computing its L_∞ -norm, *Syst. Contr. Lett.*, vol. 15, pp. 1-7, 1990.
6. S. Boyd and L. El Ghaoui, E. Feron and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, vol. 15 of “Studies in Applied Mathematics”, SIAM, Philadelphia, 1994.
7. J.R. Bunch and B.N. Parlett, Direct methods for solving symmetric indefinite systems of linear equations, *SIAM J. Numer. Anal.*, vol. 8, pp. 639-655, 1971.
8. J.R. Bunch and L. Kaufman, Some stable methods for calculating inertia and solving symmetric linear systems, *Math. Comp.*, vol. 31, pp. 162-179, 1977.
9. J.R. Bunch, Analysis of the diagonal pivoting method, *SIAM J. Numer. Anal.*, vol. 8, pp. 656-680, 1971.
10. R. Byers, A bisection method for measuring the distance of a stable matrix to the unstable matrices, *SIAM J. Sci. Stat. Comput.*, vol. 9, no. 5, pp. 875-881, 1988.
11. D. Carlson and B.N. Datta, The Lyapunov matrix equation $SA + A^*S = S^*B^*BS$, *Lin. Alg. Appl.*, vol. 28, pp. 43-52, 1979a.
12. D. Carlson and B.N. Datta, On the effective computation of the inertia of a nonhermitian matrix, *Numer. Math.*, vol. 33, pp. 315-322, 1979b.
13. D. Carlson and H. Schneider, Inertia theorems for matrices: the semidefinite case, *J. Math. Anal. Appl.*, vol. 6, pp. 430-446, 1963.
14. C.-T. Chen, A generalization of the inertia theorem, *SIAM J. Appl. Math.*, vol. 25, pp. 158-161, 1973.
15. C.-T. Chen, *Linear Systems Theory and Design*, College Publishing, New York, 1984.

16. B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, Pacific Grove, CA, 1995.
17. B.N. Datta, On the Routh-Hurwitz-Fujiwara and the Schur-Cohn-Fujiwara theorems for the root-separation problems, *Lin. Alg. Appl.*, vol. 22, pp. 135-141, 1978a.
18. B.N. Datta, An elementary proof of the stability criterion of Liénard and Chipart, *Lin. Alg. Appl.*, vol. 122, pp. 89-96, 1978b.
19. B.N. Datta, Applications of Hankel matrices of Markov parameters to the solutions of the Routh-Hurwitz and the Schur-Cohn problems, *J. Math. Anal. Appl.*, vol. 69, pp. 276-290, 1979.
20. B.N. Datta, Matrix equations, matrix polynomial, and the number of zeros of a polynomial inside the unit circle, *Lin. Multilin. Alg.* vol. 9, pp. 63-68, 1980.
21. B.N. Datta, Stability and Inertia, *Lin. Alg. Appl.* vol. 302/303, pp. 563-600, 1999.
22. B.N. Datta, and Datta, K., On finding eigenvalue distribution of a matrix in several regions of the complex plane, *IEEE Trans. Automat. Control*, AC-31, pp. 445-447, 1986.
23. B.N. Datta, and Datta, K. The matrix equation $XA = A^T X$ and an associated algorithm for inertia and stability *Lin. Alg. Appl.*, vol. 97, pp. 103-109, 1987.
24. R.A. DeCarlo, *Linear Systems - A State Variable Approach with Numerical Implementation*, Prentice Hall, Englewood Cliffs, NJ, 1989.
25. J.W. Demmel, A Counterexample for two conjectures about stability, *IEEE Trans. Automat. Contr.*, AC-32, pp. 340-342, 1987.
26. P. Dorato and R.K. Yedavalli (Editors), *Recent Advances in Robust Control*, IEEE Press, New York, NY, 1989.
27. F.R. Gantmacher, *The Theory of Matrices*, Vol. I and Vol. II., Chelsea, New York, 1959.
28. G.H. Golub, and C.F. Van Loan, *Matrix Computations*, 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.
29. K. Glover, All optimal Hankel-norm approximation of linear multivariable systems and their L_∞ error bounds, *Int. J. Control.*, vol. 39, pp. 1115-1193, 1984.
30. M. Green and D.J. Limebeer, *Linear Robust Control*, Prentice Hall Information and Systems Sciences Series, Prentice Hall, NJ, 1995, (Thomas Kailath, Editor).
31. S. Gutman, *Root Clustering in Parameter Space*, Lecture Notes in Control at Information Sciences, Springer-Verlag, New York, 1990.

32. W. Hahn, Eine Bemerkung zur zweiten methode von Lyapunov, *Math. Nachr.* vol. 14, pp. 349-354, 1955.
33. G.A. Hewer and C.S. Kenney, The sensitivity of stable Lyapunov equations, *SIAM J. Contr. and Optimiz.*, vol. 26, pp. 321-344, 1988.
34. D. Hinrichsen and B. Martensson (Editors), *Control of Uncertain Systems*, Birkhauser, Berlin, 1990.
35. D. Hinrichsen and A.J. Pritchard, Real and complex stability radii: a survey, *Control of Uncertain Systems*, Birkhauser, Berlin, 1990, (D. Hinrichsen and B. Martensson, Editors).
36. D. Hinrichsen and A.J. Pritchard, Stability radii of linear systems, *Syst. Contr. Lett.*, vol. 7, pp. 1-10, 1986.
37. R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
38. V.L. Kharitonov, Asymptotic stability of an equilibrium position of a family of linear differential equations, *Differential Equations*, vol. 14, pp. 2086-2088, 1978.
39. T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.
40. E. Kaszkurewicz and A. Bhaya, *Matrix Diagonal Stability in Systems and Computations*, Birkhauser, Boston, 1999.
41. L.H. Keel, S.P. Bhattacharyya and J.W. Howze, *Robust Control With Structured Perturbations*, *IEEE Trans. Automat. Control*, vol. 33, pp. 68-78, 1988.
42. J.P. LaSalle and S. Lefschetz, *Stability by Lyapunov's Direct Method with Applications*, Academic Press, New York, 1961.
43. S.H. Lehnigk, *Stability Theorems for Linear motions with an Introduction to Lyapunov's Direct Method*, Prentice Hall, Englewood Cliffs, NJ, 1966.
44. L.Z. Lu, A direct method for the solution of the unit circle problem, 1987 (unpublished manuscript).
45. D.G. Luenberger, *Introduction to Dynamic Systems: Theory, Models, and Applications*, John Wiley & Sons, New York, 1979.
46. A.M. Liapunov, Problème général de la stabilité du mouvement (i) *Comm. Math. Soc. Kharkov*; (1892) (ii) *Ann. Fac. Sci., Toulouse* 9 (1907); (iii) *Annals of Mathematical Studies*, 17, 1947; (iv) *Princeton University Press*, Princeton, NJ, 1949.
47. A. Malyshev and M. Sadkane, On the stability of large matrices, *J. Comput. Appl. Math.*, vol. 102, pp. 303-313, 1999.

48. M. Marden, *Geometry of Polynomials*, Amer. Math. Soc., Providence, RI, 1966.
49. A. Ostrowski and H. Schneider, Some theorems on the inertia of general matrices, *J. Math. Anal. Appl.*, vol. 4, pp. 72-84, 1962.
50. R.V. Patel and M. Toda, Quantitative measures of robustness for multivariable systems, *Proc. Amer. Control Conference*, San Francisco, 1980.
51. P. Petkov, N.D. Christov, and M.M. Konstantinov, *Computational Methods for Linear Control Systems*, Prentice Hall, London, 1991.
52. D. Pierce, *A Computational Comparison of Four Methods which Compute the Inertia of a General Matrix*, M. Sc. Thesis, Northern Illinois University, DeKalb, IL, 1983.
53. L. Qiu, B. Bernhardsson, B. Rantzer, E.J. Davison, P.M. Young, and J.C. Doyle, A formula for computation of the real stability radius, *Automatica*, vol. 31, pp. 879-890, 1995.
54. W. Rudin, *Real and Complex Analysis*, McGraw Hill, New York, 1966.
55. O. Taussky, Matrices C with $C^n \rightarrow 0$, *J. Algebra* vol. 1 pp. 5-10, 1964.
56. O. Taussky, A generalization of a theorem of Lyapunov, *J. Soc. Ind. Appl. Math.*, vol. 9, pp. 640-643, 1961.
57. C.F. Van Loan, How near is a stable matrix to an unstable matrix, *Contemporary Math.*, Amer. Math. Soc., Providence, RI, vol. 47, pp. 465-477, 1985, (R. Brualdi, et al., Editors).
58. J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.
59. H.K. Wimmer, On the Ostrowski-Schneider inertia theorem, *J. Math. Anal. Appl.* vol. 41, pp. 164-169, 1973.
60. H.K. Wimmer, Inertia theorems for matrices, controllability, and linear vibrations, *Lin. Alg. Appl.*, vol. 8, pp. 337-343, 1974.
61. H.K. Wimmer, A.D. Ziebur, Remarks on inertia theorems for matrices, *Czech. Math. J.* vol. 25, pp. 556-561, 1975.
62. W.M. Wonham, *Linear Multivariable Systems*, Springer-Verlag, New York, 1986.
63. R.K. Yedavalli, Improved measures of stability robustness for linear state space models, *IEEE Trans. Automat. Control*, AC-30, pp. 577-579, 1985.
64. K. Zhou, J.C. Doyle and K. Glover, *Robust Optimal Control*, Prentice Hall, Upper Saddle River, NJ, 1996.

65. K. Zhou and P.P. Khargonekar, Stability robustness for linear state-space models with structured uncertainty, *IEEE Trans. Automat. Control*, AC-32, pp. 621-623, 1987.

Chapter 8

NUMERICAL SOLUTIONS AND CONDITIONING OF LYAPUNOV AND SYLVESTER EQUATIONS

Contents

8.1	Introduction	291
8.2	The Existence and Uniqueness of Solutions	292
8.2.1	The Sylvester Equation: $XA + BX = C$	292
8.2.2	The Lyapunov Equation: $XA + A^T X = C$	293
8.2.3	The Discrete Lyapunov Equation: $A^T X A - X = C$	293
8.3	Perturbation Analysis and the Condition Numbers	294
8.3.1	Perturbation Analysis for the Sylvester Equation	294
8.3.2	The Condition Number of the Sylvester Equation	296
8.3.3	Perturbation Analysis for the Lyapunov Equation	297
8.3.4	The Condition Number of the Lyapunov Equation	297
8.3.5	Sensitivity of the Stable Lyapunov Equation	298
8.3.6	Sensitivity of the Discrete Lyapunov Equation	301
8.3.7	Sensitivity of the Stable Discrete Lyapunov Equation	301
8.3.8	Determining Ill-Conditioning from the Eigenvalues	302
8.3.9	A Condition Number Estimator for the Sylvester Equation: $A^T X - XB = C$	304
8.4	Analytical Methods for the Lyapunov Equations: Explicit Expressions for Solutions	306
8.5	Numerical Methods for the Lyapunov and Sylvester Equations.	307
8.5.1	Numerical Instability of Diagonalization, Jordan Canonical Form, and Companion Form Techniques	308

8.5.2	The Schur Method for the Lyapunov Equation: $XA + A^T X = C$	309
8.5.3	The Hessenberg-Schur Method for the Sylvester Equation	313
8.5.4	The Schur Method for the Discrete Lyapunov Equation	318
8.5.5	Residual and Backward Error in the Schur and Hessenberg-Schur Algorithms	321
8.5.6	A Hessenberg Method for the Sylvester Equation: $AX + XB = C$	323
8.5.7	The Hessenberg-Schur Method for the Discrete Sylvester Equation	327
8.6	Direct Computations of the Cholesky Factors of Symmetric Positive Definite Solutions of Lyapunov Equations	328
8.6.1	Computing the Cholesky Factor of the Positive Definite Solution of the Lyapunov Equation	328
8.6.2	Computing the Cholesky Factor of the Positive Definite Solution of the Discrete Lyapunov Equation	333
8.7	Comparisons of Different Methods and Conclusions	336
8.8	Some Selected Software	337
8.8.1	MATLAB CONTROL SYSTEM TOOLBOX	337
8.8.2	MATCONTROL	337
8.8.3	CSP-ANM	337
8.8.4	SLICOT	338
8.8.5	MATRIX _X	338
8.8.6	LAPACK	339
8.9	Summary and Review	339
8.10	Chapter Notes and Further Reading	340

Topics Covered

- Existence and Uniqueness Results for Lyapunov and Sylvester Equations
- Perturbation Analyses and Condition Numbers
- The Schur and the Hessenberg-Schur Methods (Both Continuous and Discrete-time Cases)
- Backward Error Analyses of the Schur and the Hessenberg-Schur Methods
- Direct Computations of Cholesky Factors of Symmetric Positive Definite Solutions of Lyapunov Equations

8.1 Introduction

As we have seen, the Lyapunov equations arise in **stability** and **robust stability** analyses, in determining **controllability** and **observability Grammians**, and in **computing H_2 -norm**. Other important applications include **internal balancing** and **model reduction** (**Chapter 9**). The solutions of Lyapunov equations are also needed for implementations of some **iterative methods for solving algebraic Riccati equations**, such as Newton's methods (**Chapter 13**). The important role of Lyapunov equations in these practical applications, warrants discussions of numerically viable techniques for their solutions.

The continuous-time **Lyapunov equation**

$$XA + A^T X = C \quad (8.1.1)$$

is a special case of another classical matrix equation, known as the **Sylvester equation**

$$XA + BX = C. \quad (8.1.2)$$

Similarly, the discrete-time Lyapunov equation

$$A^T X A - X = C$$

is a special case of the discrete-time Sylvester equation

$$BXA - X = C.$$

The Sylvester equations also arise in a wide variety of applications. For example, we will see in **Chapter 12** that a variation of the Sylvester equation, known as the **Sylvester-observer** equation, arises in the construction of **observers** and in solutions of the **eigenvalue assignment** (or pole placement) problems. The Sylvester equation also arises in other areas of applied mathematics. For example, the solution of elliptic boundary value problems can be formulated in terms of the solution of the Sylvester equation (Starke and Niethammer (1991)). The solution of the Sylvester equation is also needed in the block diagonalization of a matrix by a similarity transformation (see Datta (1995)). Once a matrix is transformed to a block diagonal form using a similarity transformation, the block diagonal form can then be conveniently used to compute the matrix exponential e^{At} .

In this Chapter, we will first develop the basic theories on the **existence** and **uniqueness** of solutions of the Sylvester and Lyapunov equations (**Section 8.2**), next discuss **perturbation theories** (**Section 8.3**), and then finally describe **computational methods** (**Sections 8.5 and 8.6**).

The continuous-time Lyapunov equation (8.1.1) and the continuous-time Sylvester equation (8.1.2) will be referred to as just the **Lyapunov** and **Sylvester equations**, respectively.

The following methods have been discussed in this chapter. They have excellent numerical properties and are recommended for use in practice:

- The **Schur methods** for the Lyapunov equations (**Sections 8.5.2 and 8.5.4**).
- The **Hessenberg-Schur Method** for the Sylvester equations (**Algorithm 8.5.1 and Section 8.5.7**).
- The **modified Schur methods** for the Cholesky factors of the Lyapunov equations (**Algorithms 8.6.1 and 8.6.2**).

Besides, a **Hessenberg method** (method based on Hessenberg decomposition only) for the Sylvester equation $AX + XB = C$ has been described in **Section 8.5.6**. The method is more efficient than the Hessenberg-Schur method, but numerical stability of this method has not been investigated yet. At present, the method is mostly of theoretical interest only.

Because of possible numerical instabilities, solving the Lyapunov and Sylvester equations via the Jordan canonical form or a companion form of the matrix A cannot be recommended for use in practice (See discussions in **Section 8.5.1**).

8.2 The Existence and Uniqueness of Solutions

In most numerical methods for solving matrix equations, it is implicitly assumed that the equation to be solved has a unique solution, and the methods then construct the unique solution. Thus, the results on the existence and uniqueness of solutions of the Sylvester and Lyapunov equations are of importance. We present some of these results in this section.

8.2.1 The Sylvester Equation: $XA + BX = C$

Assume that the matrices A , B , and C are of dimensions $n \times n$, $m \times m$, and $m \times n$, respectively. Then the following is the fundamental result on the existence and uniqueness of the Sylvester equation solution.

Theorem 8.2.1 (Uniqueness of the Sylvester Equation Solution). *Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of A , and μ_1, \dots, μ_m , be the eigenvalues of B . Then the Sylvester equation (8.1.2) has a unique solution X if and only if $\lambda_i + \mu_j \neq 0$ for all $i = 1, \dots, n$ and $j = 1, \dots, m$. In other words, the Sylvester equation has a unique solution if and only if A and $-B$ do not have a common eigenvalue.*

Proof: The Sylvester equation $XA + BX = C$ is equivalent to the $nm \times nm$ linear system

$$Px = c, \quad (8.2.1)$$

where $P = (I_n \otimes B) + (A^T \otimes I_m)$,

$$x = \text{vec}(X) = (x_{11}, \dots, x_{m1}, x_{12}, x_{22}, \dots, x_{m2}, \dots, x_{1n}, x_{2n}, \dots, x_{mn})^T,$$

$$c = \text{vec}(C) = (c_{11}, \dots, c_{m1}, c_{12}, c_{22}, \dots, c_{m2}, \dots, c_{1n}, c_{2n}, \dots, c_{mn})^T.$$

Here $W \otimes Z$ is the **Kronecker product** of W and Z . Recall from Chapter 2 that if $W = (w_{ij})$ and $Z = (z_{ij})$ are two matrices of orders $p \times p$ and $r \times r$, respectively, then their Kronecker product $W \otimes Z$ is defined by

$$W \otimes Z = \begin{pmatrix} w_{11}Z & w_{12}Z & \cdots & w_{1p}Z \\ w_{21}Z & w_{22}Z & \cdots & w_{2p}Z \\ \vdots & & & \\ w_{p1}Z & w_{p2}Z & \cdots & w_{pp}Z \end{pmatrix}. \quad (8.2.2)$$

Thus the Sylvester equation (8.1.2) has a unique solution if and only if the matrix P of the system (8.2.1) is nonsingular.

Now, the eigenvalues of the matrix P are the nm numbers $\lambda_i + \mu_j$, where $i = 1, \dots, n$ and $j = 1, \dots, m$ (Horn and Johnson (1991)). Since the determinant of a matrix is equal to the product of its eigenvalues, this means that P is nonsingular if and only if $\lambda_i + \mu_j \neq 0$, for $i = 1, \dots, n$, and $j = 1, \dots, m$. ■

8.2.2 The Lyapunov Equation: $XA + A^T X = C$

Since the Lyapunov equation (8.1.1) is a special case of the Sylvester (8.1.2) equation, the following corollary is immediate.

Corollary 8.2.1 (Uniqueness of the Lyapunov Equation Solution). *Let $\lambda_1, \lambda_2, \dots, \lambda_n$ be the eigenvalues of A . Then the Lyapunov equation (8.1.1) has a unique solution X if and only if $\lambda_i + \lambda_j \neq 0$, $i = 1, \dots, n$; $j = 1, \dots, n$.*

8.2.3 The Discrete Lyapunov Equation: $A^T X A - X = C$

The following result on the uniqueness of the solution X of the discrete Lyapunov equation

$$A^T X A - X = C \quad (8.2.3)$$

can be established in the same way as in the proof of Theorem 8.2.1.

Theorem 8.2.2 (Uniqueness of the Discrete Lyapunov Equation Solution).

Let $\lambda_1, \dots, \lambda_n$ be the n eigenvalues of A . Then the discrete Lyapunov equation (8.2.3) has a unique solution X if and only if $\lambda_i \lambda_j \neq 1$, $i = 1, \dots, n$; $j = 1, \dots, n$.

Remarks: In the above theorems, we have given results only for the uniqueness of solutions of the Sylvester and Lyapunov equations. However, there are certain control problems such as the **construction of Luenberger observer** and the **eigenvalue assignment problems**, etc.,

that require **nonsingular or full-rank solutions of the Sylvester equations** (see **Chapter 12**).

The nonsingularity of the unique solution of the Sylvester equation has been completely characterized recently by Datta, Hong, and Lee (1997). Also, partial results on nonsingularity of the Sylvester equation were obtained earlier by DeSouza and Bhattacharyya (1981), and Hearon (1977). **We will state these results in Chapter 12.**

8.3 Perturbation Analysis and the Condition Numbers

8.3.1 Perturbation Analysis for the Sylvester Equation

In this section, we study perturbation analysis of the Sylvester and Lyapunov equations and identify the condition numbers for these problems from the perturbation analysis. The results are important in assessing the accuracy of the solution obtained by a numerical algorithm. We also present an algorithm (**Algorithm 8.3.1**) for estimating the sep function that arises in computing the condition number of the Sylvester equation.

Let $\Delta A, \Delta B, \Delta C$ and ΔX be the perturbations, respectively, in the matrices A, B, C and X . Let \hat{X} be the solution of the perturbed problem. That is, \hat{X} satisfies

$$\hat{X}(A + \Delta A) + (B + \Delta B)\hat{X} = C + \Delta C. \quad (8.3.1)$$

Let

$$\epsilon = \max \left\{ \frac{\|\Delta A\|_F}{\|A\|_F}, \frac{\|\Delta B\|_F}{\|B\|_F}, \frac{\|\Delta C\|_F}{\|C\|_F} \right\}$$

Then, proceeding as in the case of perturbation analysis for the linear system problem applied to the system (8.2.1), the following result (see Higham (1996)) can be proved.

Theorem 8.3.1 (Perturbation Theorem for the Sylvester Equation)

Let the Sylvester equation $XA + BX = C$ have a unique solution X for $C \neq 0$.

Let

$$\epsilon = \max \left\{ \frac{\|\Delta A\|_F}{\alpha}, \frac{\|\Delta B\|_F}{\beta}, \frac{\|\Delta C\|_F}{\gamma} \right\} \quad (8.3.2)$$

where α, β , and γ are tolerances such that $\|\Delta A\|_F \leq \epsilon\alpha$, $\|\Delta B\|_F \leq \epsilon\beta$, and $\|\Delta C\|_F \leq \epsilon\gamma$.

Then

$$\frac{\|\Delta X\|_F}{\|X\|_F} = \frac{\|\hat{X} - X\|_F}{\|X\|_F} \leq \sqrt{3}\epsilon\delta \quad (8.3.3)$$

■

$$\text{where } \delta = \|P^{-1}\|_2 \frac{(\alpha + \beta)\|X\|_F + \gamma}{\|X\|_F}$$

Sep Function and its Role in Perturbation Results for the Sylvester Equation

Definition 8.3.1 The separation of two matrices A and B , denoted by $\text{sep}(A, B)$, is defined as:

$$\text{sep}(A, B) = \min_{X \neq 0} \frac{\|AX + XB\|_F}{\|X\|_F}$$

Thus, in terms of the sep function, we have

$$\|P^{-1}\|_2 = \frac{1}{\sigma_{\min}(P)} = \text{sep}(B, -A). \quad (8.3.4)$$

Using sep function, the equation (8.3.3) can be re-written as:

$$\frac{\|\Delta X\|_F}{\|X\|_F} < \sqrt{3}\epsilon \text{sep}(B, -A) \frac{(\alpha + \beta)\|X\|_F + \gamma}{\|X\|_F}.$$

The perturbation result (8.3.3) clearly depends upon the norm of the solution X . However, if the relative perturbations in A, B , and C are only of the order of the machine epsilon, then the following result, independent of $\|X\|$, due to Golub, Nash, and Van Loan (1979), can be established.

Corollary 8.3.1 Assume that the relative perturbations in A, B and C are all of the order of the machine precision μ , that is, $\|\Delta A\|_F \leq \mu \|A\|_F$, $\|\Delta B\|_F \leq \mu \|B\|_F$, and $\|\Delta C\|_F \leq \mu \|C\|_F$.

If X is a unique solution of the Sylvester equation $XA + BX = C$, C is nonzero and

$$\mu \frac{\|A\|_F + \|B\|_F}{\text{sep}(B, -A)} \leq \frac{1}{2}. \quad (8.3.5)$$

Then

$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} \leq 4\mu \frac{\|A\|_F + \|B\|_F}{\text{sep}(B, -A)}. \quad (8.3.6)$$

■

Example 8.3.1 Consider the Sylvester equation $XA + BX = C$ with

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} -0.9888 & 0 & 0 \\ 0 & -0.9777 & 0 \\ 0 & 0 & -0.9666 \end{pmatrix}.$$

Take $X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. Then

$$C = \begin{pmatrix} 0.0112 & 1.0112 & 2.0112 \\ 0.0223 & 1.0223 & 2.0223 \\ 0.0334 & 1.0334 & 2.0334 \end{pmatrix}.$$

Now, change the entry (1, 1) of A to 0.999999. Call this perturbed matrix \hat{A} . The matrices B and C remain unperturbed.

The computed solution of the perturbed problem (computed by MATLAB function **lyap**)

$$\hat{X} = \begin{pmatrix} 1.0001 & 0.9920 & 1.7039 \\ 1.0000 & 0.9980 & 1.0882 \\ 1.0000 & 0.9991 & 1.0259 \end{pmatrix}.$$

The relative error in the solution: $\frac{\|\hat{X} - X\|_F}{\|X\|_F} = 0.2366$. On the other hand, the relative error in the data: $\frac{\|A - \hat{A}\|_F}{\|A\|_F} = 4.0825 \times 10^{-7}$.

The phenomenon can be easily explained by noting that $\text{sep}(B, -A)$ is small: $\text{sep}(B, -A) = 1.4207 \times 10^{-6}$.

Verification of Theorem 8.3.1:

Take $\alpha = \|A\|_F$, $\beta = \|B\|_F$, and $\gamma = \|C\|_F$

Then

$$\epsilon = \frac{\|\hat{A} - A\|_F}{\|A\|_F} = 4.0825 \times 10^{-7} \quad (\text{Note that } \|\Delta B\| = 0 \text{ and } \|\Delta C\| = 0).$$

The right hand side of (8.3.3) is 2.7133.

Since $\frac{\|\hat{X} - X\|_F}{\|X\|_F} = 0.2366$, the inequality (8.3.3) is satisfied.

8.3.2 The Condition Number of the Sylvester Equation

The perturbation bound for the Sylvester equation given in Theorem 8.3.1 does not take into account of the Kronecker structure of the coefficient matrix P . The bound (8.3.3) can sometimes overestimate the effects of perturbations when A and B are only perturbed. A much sharper perturbation bound that exploits the Kronecker structure of P has been given by Higham (1996, pp. 318).

Specifically, the following result has been proved.

Theorem 8.3.2 Let $\epsilon = \max \left\{ \frac{\|\Delta A\|_F}{\alpha}, \frac{\|\Delta B\|_F}{\beta}, \frac{\|\Delta C\|_F}{\gamma} \right\}$, where α, β , and γ are tolerances given by $\|\Delta A\|_F \leq \epsilon\alpha$, $\|\Delta B\|_F \leq \epsilon\beta$, and $\|\Delta C\|_F \leq \epsilon\gamma$. Let ΔX denote the perturbation in the solution X of the Sylvester equation (8.1.2). Let P be defined by (8.2.1).

Then

$$\frac{\|\Delta X\|_F}{\|X\|_F} \leq \sqrt{3}\Psi\epsilon, \tag{8.3.7}$$

where

$$\Psi = \|P^{-1}[\beta(X^T \otimes I_m), \alpha(I_n \otimes X), -\gamma I_{mn}]\|_2 / \|X\|_F. \tag{8.3.8}$$

■

The bound (8.3.7) can be attained for any A, B , and C and we shall call the number Ψ the **condition number** of the Sylvester equation.

Remark. Examples can be constructed that show that the bounds (8.3.3) and (8.3.7) can differ by an arbitrary factor. For details, see Higham (1996).

MATCONTROL NOTE: The condition number given by (8.3.7)-(8.3.8) has been implemented in MATCONTROL function **condsylv**.

Example 8.3.2 We verify the results of Theorem 8.3.2 with Example 8.3.1.

Take $\alpha = \|A\| = 2.4494$. Then $\epsilon = 4.0825 \times 10^{-7}$.

The condition number is $\Psi = 1.0039 \times 10^6$.

$$\frac{\|\Delta X\|_F}{\|X\|_F} = 0.2366.$$

$$\sqrt{3}\Psi\epsilon = 0.7099.$$

Thus the inequality (8.3.7) is verified.

8.3.3 Perturbation Analysis for the Lyapunov Equation

Since the Lyapunov equation $XA + A^T X = C$ is a special case of the Sylvester equation, we immediately have the following Corollary of Theorem 8.3.1.

Corollary 8.3.2 (Perturbation Theorem for the Lyapunov Equation)

Let X be the unique solution of the Lyapunov equation $XA + A^T X = C; C \neq 0$. Let \hat{X} be the unique solution of the perturbed equation $\hat{X}(A + \Delta A) + (A + \Delta A)^T \hat{X} = C$,

$$\text{where } \|\Delta A\|_F \leq \mu \|A\|_F. \quad (8.3.9)$$

Assume that

$$\frac{\mu \|A\|_F}{\text{sep}(A^T, -A)} = \delta < \frac{1}{4}. \quad (8.3.10)$$

Then

$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} \leq 8\mu \frac{\|A\|_F}{\text{sep}(A^T, -A)}. \quad \blacksquare$$

8.3.4 The Condition Number of the Lyapunov Equation

For the Lyapunov equation, the **condition number** is (Higham (1996), pp. 319):

$$\phi = \|(I_n \otimes A^T + A^T \otimes I_n)^{-1}[\alpha((X^T \otimes I_n) + (I_n \otimes X)\Pi^T), -\gamma I_{n^2}]\|_2 / \|X\|_F,$$

where Π is the **vec-permutation matrix**, given by

$$\Pi = \sum_{i,j=1}^n (e_i e_j^T) \otimes (e_j e_i^T),$$

and α and γ are as defined as:

$$\|\Delta A\|_F \leq \epsilon\alpha, \text{ and } \Delta C = \Delta C^T \text{ with } \|\Delta C\|_F \leq \epsilon\gamma.$$

8.3.5 Sensitivity of the Stable Lyapunov Equation

While Corollary 8.3.2 shows that the sensitivity of the Lyapunov equation under the assumptions (8.3.9) and (8.3.10) depends upon $\text{sep}(A^T, -A)$, Hewer and Kenney (1988) have shown that if A is a stable matrix, then the sensitivity can be determined by means of the **2-norm** of the symmetric positive definite solution H of the equation

$$HA + A^T H = -I.$$

Specifically, the following result has been proved:

Theorem 8.3.3 (Perturbation Result for the Stable Lyapunov Equation)

Let A be stable and let X satisfy $XA + A^T X = -C$. Let ΔX and ΔC , respectively, be the perturbations in X and C such that

$$(A + \Delta A)^T(X + \Delta X) + (X + \Delta X)(A + \Delta A) = -(C + \Delta C). \quad (8.3.11)$$

Then

$$\frac{\|\Delta X\|}{\|X + \Delta X\|} \leq 2 \|A + \Delta A\| \|H\| \left[\frac{\|\Delta A\|}{\|A + \Delta A\|} + \frac{\|\Delta C\|}{\|C + \Delta C\|} \right], \quad (8.3.12)$$

where H satisfies the Lyapunov equation and $\|\cdot\|$ represents the 2-norm.

$$HA + A^T H = -I.$$

Proof: Since A is stable, we may write $H = \int_0^\infty e^{A^T t} e^{At} dt$.

Now from $XA + A^T X = -C$ and (8.3.11), we have

$$A^T \Delta X + \Delta X A = -(\Delta C + \Delta A^T(X + \Delta X) + (X + \Delta X)\Delta A). \quad (8.3.13)$$

Since (8.3.13) is a Lyapunov equation in ΔX and A is stable, we may again write

$$\Delta X = \int_0^\infty e^{A^T t} (\Delta C + \Delta A^T(X + \Delta X) + (X + \Delta X)\Delta A) e^{At} dt.$$

Let u and v be the left and right singular vectors of unit length of ΔX associated with the largest singular value. Then multiplying the above equation by u^T to the left and by v to the right, we have

$$\begin{aligned} \|\Delta X\| &= \int_0^\infty \left| u^T e^{A^T t} (\Delta C + \Delta A^T(X + \Delta X) + (X + \Delta X)\Delta A) e^{At} v \right| dt \\ &\leq \|\Delta C + \Delta A^T(X + \Delta X) + (X + \Delta X)\Delta A\| \int_0^\infty \|e^{At} u\| \|e^{At} v\| dt \\ &\leq (\|\Delta C\| + 2 \|\Delta A\| \|X + \Delta X\|) \int_0^\infty \|e^{At} u\| \|e^{At} v\| dt. \end{aligned} \quad (8.3.14)$$

Now, by the Cauchy-Schwarz inequality, we have

$$\int_0^\infty \| e^{At}u \| \| e^{At}v \| dt \leq \left[\int_0^\infty \| e^{At}u \|^2 dt \right]^{\frac{1}{2}} \left[\int_0^\infty \| e^{At}v \|^2 dt \right]^{\frac{1}{2}}$$

Again

$$\begin{aligned} \int_0^\infty \| e^{At}u \|^2 dt &= \int_0^\infty u^T e^{A^T t} e^{At} u dt \\ &= u^T \left[\int_0^\infty e^{A^T t} e^{At} dt \right] u \\ &= u^T H u, \text{ where } H = \int_0^\infty e^{A^T t} e^{At} dt \end{aligned}$$

Since $\| u \|_2 = 1$ and H is symmetric positive definite (because A is stable), we have

$$u^T H u \leq \| H \|,$$

and thus

$$\int_0^\infty \| e^{At}u \|^2 dt \leq \| H \|.$$

Similarly $\int_0^\infty \| e^{At}v \|^2 dt \leq \| H \|$.

Thus, from (8.3.14), we have

$$\| \Delta X \| \leq (\| \Delta C \| + 2 \| \Delta A \| \| X + \Delta X \|) \| H \| . \quad (8.3.15)$$

Again from (8.3.11), we have

$$\| C + \Delta C \| \leq 2 \| A + \Delta A \| \| X + \Delta X \| . \quad (8.3.16)$$

Combining of (8.3.15) with (8.3.16) we obtain the desired result. ■

Remark: The results of Theorem 8.3.3 hold for any perturbation.

In particular, if

$$\| \Delta C \| \leq \mu \| C \|, \| \Delta A \| \leq \mu \| A \|,$$

and $8\mu \| A \| \| H \| \leq \frac{1-\mu}{1+\mu}$,

then it can be shown that

$$\frac{\| \Delta X \|}{\| X \|} \leq 8\mu \| A \| \| H \| (1-\mu) \approx 8\mu \| A \| \| H \| . \quad (8.3.17)$$

Example 8.3.3 Consider the Lyapunov equation (8.1.1) with

$$A = \begin{pmatrix} -1 & 2 & 3 \\ 0 & -0.0001 & 3 \\ 0 & 0 & -3 \end{pmatrix} \text{ and } C = \begin{pmatrix} -2 & 0.9999 & 2 \\ 0.9999 & 3.9998 & 4.9999 \\ 2 & 4.9999 & 6 \end{pmatrix}.$$

A is stable. The exact solution X of the Lyapunov equation $XA + A^T X = C$ is

$$X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

The solution H of the Lyapunov equation $HA + A^T H = -I$ is

$$H = 10^4 \begin{pmatrix} 0.0001 & 0.0001 & 0.0001 \\ 0.0001 & 2.4998 & 2.4999 \\ 0.0001 & 2.4999 & 2.5000 \end{pmatrix}.$$

Since $\|H\| = 4.9998 \times 10^4$ and $\|A\| = 5.3744$, according to Theorem 8.3.3, the Lyapunov equation with above A and C is expected to be ill-conditioned. We verify this as follows:

Perturb the (1, 1) entry of A to -0.9999999 and keep the other entries of A and those of C unchanged. Then the computed solution \hat{X} with this slightly perturbed A is

$$\hat{X} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1.006 & 1.006 \\ 1 & 1.006 & 1.006 \end{pmatrix}.$$

Note that if \hat{A} denotes the perturbed A , then

the relative perturbation in A : $\frac{\|\hat{A} - A\|}{\|A\|} = 1.8607 \times 10^{-8}$. On the other hand, the relative change in the solution X : $\frac{\|\hat{X} - X\|}{\|X\|} = 0.0040$.

Verification of the Result of Theorem 8.3.3

$$\frac{\|\Delta X\|}{\|X + \Delta X\|} = 0.0040, \quad \Delta C = 0.$$

$$2\|A + \Delta A\|\|H\| \frac{\|\Delta A\|}{\|A + \Delta A\|} = 0.0100$$

Thus the inequality (8.3.12) is satisfied.

Verification of the inequality (8.3.17)

Since $\frac{\|\Delta A\|}{\|A\|} = 1.8607 \times 10^{-8}$, we take $\mu = 1.8607 \times 10^{-8}$.

Then $8\mu\|A\|\|H\| = 0.04 \leq \frac{1-\mu}{1+\mu} = 0.9999996$. Thus the hypothesis is satisfied.

$$\text{Also, } \frac{\|\Delta X\|}{\|X\|} = 0.004, \quad 8\mu\|A\|\|H\| = 0.04$$

Thus the inequality (8.3.17) is satisfied.

8.3.6 Sensitivity of the Discrete Lyapunov Equation

Consider now the discrete Lyapunov equation:

$$A^T X A - X = C$$

This equation is equivalent to the linear system:

$$Rx = c$$

where $R = A^T \otimes A^T - I_{n^2}$ (I_{n^2} is the $n^2 \times n^2$ identity matrix).

Applying the results of perturbation analysis to the linear system $Rx = c$, the following result can be proved.

Theorem 8.3.4 (Perturbation Result for the Discrete Lyapunov Equation)

Let X be the unique solution of the discrete Lyapunov equation

$$A^T X A - X = C.$$

Let \hat{X} be the unique solution of the perturbed equation where the perturbation in A is of order machine precision μ .

Assume that

$$\frac{(2\mu + \mu^2) \|A\|_F^2}{\text{sep}_d(A^T, A)} = \delta < 1,$$

where $\text{sep}_d(A^T, A) = \min_{x \neq 0} \frac{\|Rx\|_2}{\|x\|_2} = \min_{X \neq 0} \frac{\|A^T X A - X\|_F}{\|X\|_F} = \sigma_{\min}(A^T \otimes A^T - I_{n^2})$.

Then

$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} \leq \frac{\mu}{1 - \delta} \frac{(3 + \mu) \|A\|_F^2 + 1}{\text{sep}_d(A^T, A)}. \quad (8.3.18)$$

■

8.3.7 Sensitivity of the Stable Discrete Lyapunov Equation

As in the case of the stable Lyapunov equation, it can be shown [Gahinet, Laub, Kenney, and Hewer (1990)] that the sensitivity of the stable discrete Lyapunov equation can also be measured by the **2-norm** of the unique solution of the discrete Lyapunov equation: $A^T X A - X = -I$. Specifically, the following result has been proved by Gahinet, Laub, Kenney and Hewer (1990).

Theorem 8.3.5 (Sensitivity of the Stable Discrete Lyapunov Equation)

Let A be discrete stable. Let H be the unique solution of

$$A^T H A - H = -I,$$

then $\text{sep}_d(A^T, A) \geq \frac{\sqrt{n}}{\|H\|_2}$. ■

Example 8.3.4 Let

$$A = \begin{pmatrix} 0.9990 & 1 & 1 \\ 0 & 0.5000 & 1 \\ 0 & 0 & 0.8999 \end{pmatrix}.$$

Set

$$X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \text{ and take } C = A^T X A - X. \text{ Then}$$

$$H = 10^5 \begin{pmatrix} 0.0050 & 0.0100 & 0.1482 \\ 0.0100 & 0.0200 & 0.2970 \\ 0.1482 & 0.2970 & 4.4504 \end{pmatrix},$$

$$\| H \|_2 = 4.4752 \times 10^5.$$

By Theorem 8.3.5, the discrete Lyapunov equation $A^T X A - X = C$ is expected to be ill-conditioned. We verify this as follows.

Let $a(2, 2)$ be perturbed to 0.4990 and all other entries of A and of C remain unchanged. Let \hat{A} denote the perturbed A . Let \hat{X} be the solution of the perturbed problem. Then \hat{X} , computed by the MATLAB function **dlyap**, is

$$\hat{X} = \begin{pmatrix} 1 & 1 & 1.0010 \\ 1 & 1 & 1.0019 \\ 1.0010 & 1.0019 & 1.0304 \end{pmatrix}.$$

The relative error in X :

$$\frac{\| \hat{X} - X \|_2}{\| X \|_2} = 0.0102.$$

The relative perturbation in A :

$$\frac{\| \hat{A} - A \|_2}{\| A \|_2} = 4.8244 \times 10^{-5}.$$

8.3.8 Determining Ill-Conditioning from the Eigenvalues

Since $\| P^{-1} \|_2 = \frac{1}{\text{sep}(B, -A)}$ is not easily computable, and $\text{sep}(B, -A) > 0$ if and only if B and $-A$ have no common eigenvalues, one may wonder if the ill-conditioning of P^{-1} (and therefore of the Sylvester or the Lyapunov equation) can be determined a priori from the eigenvalues of A and B .

The following result can be easily proved to this effect (Ghavimi and Laub (1995)).

Theorem 8.3.6 The Sylvester equation $XA + BX = C$ is ill-conditioned if both coefficient matrices A and B are ill-conditioned with respect to inversion.

Example 8.3.5 Let $A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0.001 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 0.0001 & 1 \\ 0 & 0 & 0.0001 \end{pmatrix}$ and

$$C = \begin{pmatrix} 8 & 8 & 8.001 \\ 3.0001 & 3.0001 & 3.0011 \\ 2.0001 & 2.0001 & 2.0011 \end{pmatrix}.$$

The exact solution is $X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

Now change $a(3,1)$ to 0.99999 and keep the rest of the data unchanged. Then the solution of

the perturbed problem is $\hat{X} = \begin{pmatrix} 908.1970 & -905.2944 & -906.2015 \\ -452.6722 & 454.2208 & 454.6745 \\ 1.0476 & 0.9524 & 0.9524 \end{pmatrix}$,

which is completely different from the exact solution X .

Note that the relative error in the solution: $\frac{\|X - \hat{X}\|}{\|X\|} = 585.4190$.

However, the relative perturbation in the data: $\frac{\|A - \hat{A}\|}{\|A\|} = 4.5964 \times 10^{-6}$ (\hat{A} is the perturbed matrix). This drastic change in the solution X can be explained by noting that **A and B are both ill-conditioned**:

$$\begin{aligned} \text{Cond}(A) &= 6.1918 \times 10^{16} \\ \text{Cond}(B) &= 8.5602 \times 10^8 \end{aligned}$$

Remarks: The converse of the above Theorem is, in general, not true.

To see this, consider Example 8.3.1 once more.

We have seen that the Sylvester equation with the data of this example is ill-conditioned.

But note that $\text{Cond}(A) = 4.0489$, $\text{Cond}(B) = 0.9888$. Thus neither A nor B is ill-conditioned.

Near singularity of A and the Ill-conditioning of the Lyapunov Equation

From Theorem 8.3.6 we immediately obtain the following corollary:

Corollary 8.3.3 If A is nearly singular, then the Lyapunov equation $XA + A^T X = C$ is ill-conditioned.

Example 8.3.6 Let

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0.0001 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 2 & 2.0001 & 4 \\ 2.0001 & 2.0002 & 4.0001 \\ 4 & 4.0001 & 6 \end{pmatrix}.$$

The exact solution $X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. Now perturb the $(1, 1)$ entry of A to 0.9880. Call the perturbed matrix \hat{A} . The computed solution of the perturbed problem

$$\hat{X} = \begin{pmatrix} 1.0121 & 0.9999 & 1.0000 \\ 0.9999 & 2.4750 & -0.4747 \\ 1.000 & -0.4747 & 2.4747 \end{pmatrix}.$$

The relative error in X : $\frac{\|\hat{X} - X\|}{\|X\|} = 0.9832$.

The relative perturbation in A : $\frac{\|\hat{A} - A\|}{\|A\|} = 0.0060$.

The ill-conditioning of the Lyapunov equation with the given data can be explained from the fact that A is nearly singular. Note that $\text{Cond}(A) = 3.9999 \times 10^4$ and $\text{sep}(A^T, -A) = 5.001 \times 10^{-5}$.

8.3.9 A Condition Number Estimator for the Sylvester Equation: $A^T X - XB = C$

We have seen in **Section 8.3.1** that

$$\text{sep}(B, -A) = \frac{1}{\|P^{-1}\|_2} = \frac{1}{\sigma_{\min}(P)},$$

where P is the coefficient matrix of the linear system (8.2.1).

However, finding $\text{sep}(B, -A)$ by computing the smallest singular value of P^{-1} requires a major computational effort. Even for modest m and n , it might be computationally prohibitive from the viewpoints of both the storage and the computational cost. It will require $O(m^3n^3)$ flops and $O(m^2n^2)$ storage.

Byers (1984) has proposed an algorithm to estimate $\text{sep}(A, B^T)$ in the style of the LINPACK condition number estimator. The LINPACK condition number estimator for $\text{Cond}(P)$ is based on estimating $\|P^{-1}\|_2$ by $\|y\|/\|z\|$, where y , z , and w satisfy

$$\begin{aligned} P^T z &= w \\ Py &= z; \end{aligned}$$

the components of the vector w are taken to be $w_i = \pm 1$, where the signs are chosen such that the growth in z is maximized.

Algorithm 8.3.1 *Estimating $\text{sep}(A, B^T)$.*

Input: $A_{m \times m}, B_{n \times n}$ – Both upper triangular matrices.

Output: *Sepest - An estimate of $\text{sep}(A, B^T)$.*

Step 1.

```

For  $i = m, m-1, \dots, 1$  do
  For  $j = n, n-1, \dots, 1$  do
     $p \equiv \left( \sum_{h=i+1}^m a_{ih} z_{hj} - \sum_{h=j+1}^n z_{ih} b_{jh} \right)$ 
     $w \equiv -\text{sign}(p)$ 
     $z_{ij} \equiv (w - p) / (a_{ii} - b_{jj})$ 
  End
End

```

Step 2. Compute $Z \equiv Z/\|Z\|$, where $Z = (z_{ij})$.

Step 3. Solve for Y : $A^T Y - Y B = Z$.

Step 4. $Sepest = 1/\|Y\|$.

Example 8.3.7 Consider estimating $\text{sep}(B, -A)$ with

$$A = \begin{pmatrix} -1 & 2 & 3 \\ 0 & -2 & 1 \\ 0 & 0 & 0.9990 \end{pmatrix} \text{ and } B = \begin{pmatrix} -1 & 2 & 3 \\ 0 & -2.5 & 0 \\ 0 & 0 & 1.9999 \end{pmatrix}.$$

The algorithm produces $\text{sepest}(B, -A) = 3.0340 \times 10^{-5}$, whereas the actual value of $\text{sep}(B, -A) = 3.0263 \times 10^{-5}$.

Remarks:

1. If $p = 0$, $\text{sign}(p)$ can be taken arbitrarily.
2. The major work in the algorithm is in the solution of the Sylvester equation in Step 3. Thus, once this equation is solved, the remaining part of the algorithm requires only a little extra work. Efficient numerical methods for solving the Sylvester and Lyapunov equations are discussed in the next section.

Flop-count. The algorithm requires $2(m^2n + mn^2)$ flops.

Remarks:

1. The algorithm must be modified when A and B are in quasi-triangular forms (real Schur forms), or one is in Hessenberg form and the other is in real Schur form.
2. There exists a LAPACK-style estimation for $\text{sep}(B, -A)$. For details, see Kågström and Poromaa (1989, 1992, 1996).

MATCONTROL NOTES: The `sep` function can be computed using the Kronecker product in MATCONTROL function `sepkr`. Algorithm 8.3.1 has been implemented in MATCONTROL function `sepest`, which calls the function `sylvhutc` for solving the upper triangular Sylvester equation in Step 3.

8.4 Analytical Methods for the Lyapunov Equations: Explicit Expressions for Solutions

There are numerous methods for solving Lyapunov and Sylvester equations. They can be broadly classified into two classes: **Analytical** and **Numerical** Methods.

By an analytical method, we mean a method that attempts to give an explicit expression for the solution matrix (usually the unique solution).

Recall from Chapter 7 that when A is a stable matrix, a unique solution X of the Lyapunov equation

$$XA + A^T X = -C$$

is given by

$$X = \int_0^\infty e^{A^T t} C e^{At} dt.$$

Similarly, when A is discrete-stable, a unique solution X of the discrete Lyapunov equation

$$A^T X A - X = C$$

is given by

$$X = - \sum_{k=0}^{\infty} (A^k)^T C A^k. \quad (8.4.1)$$

Thus to solve the continuous-time Lyapunov equation, we need to compute the exponential matrix e^{At} and evaluate a matrix integral. In the discrete-case, various powers of A need to be computed and many matrix multiplications need to be performed. We have already seen that there are some obvious computational difficulties with these computations.

The other analytical methods include **finite and infinite series methods** (see Barnett and Storey (1970)).

These methods again have some severe computational difficulties. For example, consider the solution of the Lyapunov equation

$$XA + A^T X = C,$$

using the **finite series method** proposed by Jameson (1968). The method can be briefly described as follows:

Let the characteristic polynomial of A be $\det(\lambda I - A) = \lambda^n + c_1\lambda^{n-1} + \cdots + c_n$.

Define the sequence of matrices $\{Q_k\}$ by

$$\begin{aligned} Q_{-1} &= 0, & Q_0 &= C \\ Q_k &= A^T Q_{k-1} - Q_{k-1} A + A^T Q_{k-2} A, & k &= 1, 2, \dots, n. \end{aligned} \quad (8.4.2)$$

Then it has been shown that

$$X = P^{-1}(Q_n - c_1 Q_{n-1} + \cdots + (-1)^n c_n Q_0), \quad (8.4.3)$$

where $P = (A^T)^n - c_1 (A^T)^{n-1} + \cdots + (-1)^n c_n I$.

As can be seen from the description of the method that the method is not numerically effective for practical computations.

Note that for computation of the matrix P , various powers of A need to be computed and the matrix P can be ill-conditioned, which will affect the accuracy of X . This, together with the fact that the sensitivity of the characteristic polynomial of a matrix A (due to the small perturbations in the coefficients) grows as the order of the matrix grows (in general), lead us to believe that such methods will give unacceptable accuracy. **Indeed, the numerical experiments show that for random matrices of size 14×14 the errors are almost as large as the solutions themselves.**

Thus, we will not pursue further with the analytic methods. However, for reader's convenience to compare this method with other numerically viable methods, the finite series method has been implemented in MATCONTROL function `lyapfns`.

8.5 Numerical Methods for the Lyapunov and Sylvester Equations.

An obvious way to solve the Sylvester equation $XA + BX = C$ is to apply Gaussian elimination with partial pivoting to the system $Px = c$ given by (8.2.1). But, unless the special structure of P can be exploited, Gaussian elimination scheme for the Sylvester equation will be **computationally prohibitive**, since $O(n^3m^3)$ flops and $O(n^2m^2)$ storage will be required. One way to exploit the structure of P will be to transform A and B to some **simple forms** using similarity transformations.

Thus, if U and V are nonsingular matrices such that

$$U^{-1}AU = \hat{A}, \quad V^{-1}BV = \hat{B} \quad \text{and} \quad V^{-1}CU = \hat{C}$$

then $XA + BX = C$ is transformed to

$$Y\hat{A} + \hat{B}Y = \hat{C}, \quad (8.5.1)$$

where $Y = V^{-1}XU$.

If \hat{A} and \hat{B} are in simple forms, then the equation $Y\hat{A} + \hat{B}Y = \hat{C}$ can be easily solved, and the solution X can then be recovered from Y . The idea, therefore, is summarized in the following steps:

Step 1. Transform A and B to “**simple**” forms, using similarity transformations:

$$\hat{A} = U^{-1}AU, \quad \hat{B} = V^{-1}BV$$

Step 2. Update the right hand side matrix: $\hat{C} = V^{-1}CU$

Step 3. Solve the **transformed equation** for Y : $Y\hat{A} + \hat{B}Y = \hat{C}$

Step 4. Recover X from Y by solving the system: $XU = VY$.

The usual “simple” forms, discussed earlier in Chapter 4 and elsewhere in this book, include:

1. **Diagonal Forms**
2. **Companion Forms**
3. **Jordan Canonical Forms**
4. **Hessenberg Forms**
5. **Real Schur Forms**

8.5.1 Numerical Instability of Diagonalization, Jordan Canonical Form, and Companion Form Techniques.

It is true that the rich structures of Jordan and companion matrices can be nicely exploited in solving the reduced Sylvester equation (8.5.1). However, as noted before, the companion, and Jordan Canonical forms, in general, can not be obtained in a numerically stable way. (For more on numerically computing the Jordan Canonical form, see Golub and Wilkinson (1976)). The transforming matrices will be, in some cases, ill-conditioned and this ill-conditioning will affect the computations of $\hat{A}, \hat{B}, \hat{C}$ and X (from Y), which require computations involving inverses of the transforming matrices. Indeed, **numerical experiments performed by us show that solutions of the Sylvester equation using companion form of A with A of sizes larger than 15 have errors almost as large as the solutions themselves.** We will illustrate below by a simple example how diagonalization technique yields an inaccurate solution.

Example 8.5.1 Consider solving the Lyapunov equation $XA + A^T X - C = 0$ with

$$A = \begin{pmatrix} 2.4618 & -1.5284 & 2.2096 & -0.3503 \\ 5.5854 & -1.2161 & 2.3825 & -1.2843 \\ 1.6935 & 2.5009 & 2.1131 & -1.2186 \\ -0.2686 & -3.2594 & 7.9205 & 0.6412 \end{pmatrix}.$$

Choose

$$X = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Compute

$$C = XA + A^T X = \begin{pmatrix} 18.9439 & 5.9690 & 24.0976 & 7.2599 \\ 5.9690 & -7.0060 & 11.1226 & -5.7150 \\ 24.0976 & 11.1226 & 29.2513 & 12.4136 \\ 7.2599 & -5.7150 & 12.4136 & -4.4240 \end{pmatrix}.$$

Let X_{Diag} be the solution obtained by the diagonalization procedure (using MATLAB Function **lyap2**).

The relative residual:

$$\frac{\| X_{Diag}A + A^T X_{Diag} - C \|}{\| X_{Diag} \|} = 1.6418 \times 10^{-7}$$

The solution \hat{X} obtained by MATLAB function **lyap** (based on the numerically viable **Schur method**):

$$\hat{X} = \begin{pmatrix} 1.0000 & 1.0000 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 1.0000 & 1.0000 \end{pmatrix}$$

The relative residual:

$$\frac{\| \hat{X}A + A^T \hat{X} - C \|}{\| \hat{X} \|} = 9.5815 \times 10^{-15}$$

Solutions via Hessenberg and Schur Forms

In view of the remarks made above, our “simple” forms of choice have to be **Hessenberg forms** and the (**real**) **Schur forms**; since we know that the transforming matrices U and V in these cases can be chosen to be orthogonal, which are perfectly well-conditioned. Some such methods are discussed below.

8.5.2 The Schur Method for the Lyapunov Equation: $XA + A^TX = C$

The following method, proposed by Bartels and Stewart (1972), is **now widely used as an effective computational method for the Lyapunov equation**. The method is based on reduction of A^T to real Schur form. It is, therefore, known as the **Schur method for the Lyapunov equation**. The method is described as follows:

Step 1. Reduction of the Problem. Let $R = U^T A^T U$ be the **real Schur form** of the matrix A^T . Then, employing this transformation, the Lyapunov matrix equation $XA + A^TX = C$ is reduced to

$$YR^T + RY = \hat{C}, \quad (8.5.2)$$

where $R = U^T A^T U$, $\hat{C} = U^T C U$ and $Y = U^T X U$.

Step 2. Solution of the Reduced Problem. The reduced equation to be solved is: $YR^T + RY = \hat{C}$. Let

$$Y = (y_1, \dots, y_n), \quad \hat{C} = (c_1, \dots, c_n), \quad \text{and} \quad R = (r_{ij}).$$

Assume that the columns y_{k+1} through y_n have been computed, and consider the following two cases.

Case 1: $r_{k,k-1} = 0$. Then y_k is determined by solving the quasi-triangular system

$$(R + r_{kk}I)y_k = c_k - \sum_{j=k+1}^n r_{kj}y_j.$$

If, in particular, **R is upper triangular**, that is there are no “**Schur bumps**” on the diagonal; then each $y_i, i = n, n-1, \dots, 2, 1$ can be obtained by solving an $n \times n$ upper triangular system as follows:

$$\begin{aligned} (R + r_{nn}I)y_n &= c_n \\ (R + r_{n-1,n-1}I)y_{n-1} &= c_{n-1} - r_{n-1,n}y_n \\ &\vdots \\ (R + r_{11}I)y_1 &= c_1 - r_{12}y_2 - \cdots - r_{1n}y_n \end{aligned} \tag{8.5.3}$$

Remark. If the complex Schur decomposition is used, that is if $R_c = U_c^* A^T U_c$ is a complex triangular matrix, then the solution Y_c of the reduced problem is computed by solving n complex $n \times n$ linear systems (8.5.3). The MATLAB function **rsf2csf** converts a real Schur form to a complex triangular matrix. However, the use of complex arithmetic is more expensive and **not recommended in practice**.

Case 2: $r_{k,k-1} \neq 0$ **for some k** . This indicates that there is a “**Schur bump**” on the diagonal. This enables us to compute y_{k-1} and y_k simultaneously, by solving the following $2n \times 2n$ linear system:

$$R(y_{k-1}, y_k) + (y_{k-1}, y_k) \begin{pmatrix} r_{k-1,k-1} & r_{k,k-1} \\ r_{k-1,k} & r_{kk} \end{pmatrix} = (c_{k-1}, c_k) - \sum_{j=k+1}^n (r_{k-1,j}y_j, r_{kj}y_j) = (d_{k-1}, d_k) \tag{8.5.4}$$

Remark. To distinguish between Case 1 and Case 2 in a computational setting, it is recommended that some threshold, for example, $Tol = \mu \|A\|_F$ be used, where μ is the machine precision.

Thus, to see if $r_{k,k-1} = 0$, accept $r_{k,k-1} = 0$ if $|r_{k,k-1}| < Tol$.

An Illustration

We illustrate the above procedure with $n = 3$.

Assume that $r_{21} \neq 0$, that is,

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{pmatrix}$$

Since $r_{32} = 0$, by **Case 1**, y_3 is computed by solving the system:

$$(R + r_{33}I)y_3 = c_3$$

Since $r_{21} \neq 0$, by **Case 2**, y_1 and y_2 are computed by solving

$$R(y_1, y_2) + (y_1, y_2) \begin{pmatrix} r_{11} & r_{21} \\ r_{12} & r_{22} \end{pmatrix} = (c_1 - r_{13}y_3, c_2 - r_{23}y_3) \quad (8.5.5)$$

Step 3. Recovery of the solution of the original problem from the solution of the reduced problem. Once Y is obtained by solving the reduced problem $YR^T + RY = \hat{C}$, the solution X of the original problem $XA + A^TX = C$, is recovered as

$$X = UYU^T.$$

Example 8.5.2 Consider solving the Lyapunov equation $XA + A^TX = C$ with

$$A = \begin{pmatrix} 0 & 2 & -1 \\ -3 & -2 & 2 \\ -2 & 1 & -1 \end{pmatrix} \text{ and } C = \begin{pmatrix} -2 & 2 & -3 \\ -8 & -6 & -5 \\ 11 & 13 & -2 \end{pmatrix},$$

Step 1. Reduction. Using MATLAB function $[U, R] = \text{schur}(A^T)$, we obtain

$$R = \begin{pmatrix} -1.3776 & 3.8328 & 1.3064 \\ -1.0470 & 0.8936 & -1.2166 \\ 0 & 0 & -2.5160 \end{pmatrix}, \quad U = \begin{pmatrix} 0.7052 & 0.4905 & 0.5120 \\ 0.6628 & -0.7124 & -0.2304 \\ -0.2518 & -0.5019 & 0.8275 \end{pmatrix}$$

Then $\hat{C} = U^T C U$ is

$$\hat{C} = \begin{pmatrix} -9.3174 & 1.9816 & -7.5863 \\ -2.1855 & -1.0425 & 2.4422 \\ 16.2351 & -3.4886 & 0.3600 \end{pmatrix}$$

Step 2. Solve $RY + YR^T = \hat{C}$. Since $r_{32} = 0$, then by **Case 1**, y_3 is computed by solving the system:

$$\begin{pmatrix} -3.8936 & 3.8328 & 1.3064 \\ -1.0470 & -1.6224 & -1.2166 \\ 0 & 0 & -5.0320 \end{pmatrix} y_3 = \begin{pmatrix} -7.5863 \\ 2.4422 \\ 0.3600 \end{pmatrix}.$$

$$y_3 = \begin{pmatrix} 0.3030 \\ -1.6472 \\ -0.0715 \end{pmatrix}$$

Since $r_{21} \neq 0$, then by **Case 2**, y_1 and y_2 are computed by solving the system (8.5.5):

$$(y_1^T, y_2^T) = (3.4969, 0.1669, -1.2379, 0.2345, 0.5746, 3.0027)^T.$$

Step 3. Recovery of the solution.

$$X = UYU^T = \begin{pmatrix} 2 & 0 & -2 \\ 2 & 2 & 1 \\ 0 & -3 & 0 \end{pmatrix}.$$

Example 8.5.3 We now solve the previous example (Example 8.5.2) using the **complex Schur form**

$$\text{Step 1. } R = \begin{pmatrix} -0.2420 + 1.6503i & -0.3227 + 3.5797i & -0.0927 - 0.9538i \\ 0 & -0.2420 - 1.6503i & 1.2113 - 0.8883i \\ 0 & 0 & -2.5160 \end{pmatrix},$$

$$U = \begin{pmatrix} -0.5814 + 0.5148i & 0.0802 - 0.3581i & 0.5120 \\ -0.0030 + 0.4839i & 0.6649 + 0.5202i & -0.2304 \\ 0.3590 - 0.1838i & 0.1355 + 0.3664i & -0.8275 \end{pmatrix},$$

$$\hat{C} = \begin{pmatrix} -7.5894 + 1.4093i & 1.8383 + 4.252i & 2.6799 + 5.5388i \\ 2.1139 - 1.1953i & -2.7706 - 1.4093i & -4.7410 + 1.783i \\ -6.5401 + 11.8534i & 9.2728 + 2.5470i & 0.3600 \end{pmatrix}$$

Step 2. Since R is triangular (complex), the columns y_1, y_2, y_3 of y are successively computed by solving **complex linear systems** (8.5.3). This gives

$$y_1 = (-2.9633 + 0.000229j, -0.7772 + 0.8659j, -0.7690 - 0.9038j)^T$$

$$y_2 = (-0.7811 - 0.8163j, 1.1082 - 0.0229j, -2.0819 - 2.923j)^T$$

$$y_3 = (0.6108 - 0.2212j, 0.9678 - 1.2026j, -0.0715)^T$$

Thus, with $Y = (y_1, y_2, y_3)$, we have

$$X = UYU^T = \begin{pmatrix} 2 & 0 & -2 \\ 2 & 2 & 1 \\ 0 & -3 & 0 \end{pmatrix}.$$

Note: In practice, the system (8.5.4) is solved using Gaussian elimination with partial pivoting. The LAPACK and SLICOT routines (see **Section 8.10.4**) have used Gaussian elimination with complete pivoting (see Datta (1995), and Golub and Van Loan (1996)) and the structure of the real Schur form has been exploited there. For details of implementations, the readers may consult the book by Sima (1996).

MATCONTROL NOTE: The Schur method for the Lyapunov equation has been implemented in MATCONTROL function **lyaprsc**.

MATLAB NOTE: MATLAB function **lyap** in the form

$$X = \mathbf{lyap}(A, C)$$

solves the Lyapunov equation

$$AX + XA^T = -C$$

using the **complex Schur** triangularization of A .

Flop-count.

1. Transformation of A to RSF: $26n^3$ (Assuming that the QR iteration algorithm requires about 2 iterations to make a subdiagonal entry negligible). (This count includes construction of U).
2. Solution of the reduced problem: $3n^3$
3. Recovery of the solution: $3n^3$ (using the symmetry of X).

Total Flops: $32n^3$ (**Approximate**).

8.5.3 The Hessenberg-Schur Method for the Sylvester Equation

The Schur method described above for the Lyapunov equation can also be used to solve the Sylvester equation $XA + BX = C$. The matrices A and B are, respectively, transformed to the lower and upper real Schur forms, and then back-substitution is used to solve the reduced Schur problem. Note, that the special form of the Schur matrix S can be exploited only in the solution of the $m \times m$ linear systems with S . Some computational effort can be saved if B , the larger of the two matrices A and B , is left in Hessenberg form, while the smaller matrix A is transformed further to real Schur form. The reason for this is that a matrix must be transformed to a Hessenberg matrix as an initial step in the reduction to real Schur form. The important outcome here is that back substitution for the solution of the Hessenberg-Schur problem is still possible. Noting this, Golub, Nash and Van Loan (1979) developed the following Hessenberg-Schur method for the Sylvester equation problem.

Step 1. Reduction to the Hessenberg-Schur Problem. Assume that m is larger than n . Let $R = U^T A^T U$ and $H = V^T B V$ be, respectively, the upper real Schur form and the upper Hessenberg form, of A and B . Then

$$\begin{aligned} XA + BX = C &\quad \text{becomes} \quad YR^T + HY = \hat{C}, \quad \text{where} \\ Y = V^T X U, \quad \hat{C} &= V^T C U. \end{aligned} \tag{8.5.6}$$

Step 2. Solution of the Reduced Hessenberg-Schur Problem. In the reduced problem $HY + YR^T = \hat{C}$, let $Y = (y_1, y_2, \dots, y_n)$ and $\hat{C} = (c_1, \dots, c_n)$. Then, assuming that y_{k+1}, \dots, y_n have already been computed, y_k (or y_k and y_{k+1}) can be computed as in the case of the Lyapunov equation, by considering the following two cases.

Case 1. If $r_{k,k-1} = 0$, y_k is computed by solving the $m \times m$ Hessenberg system:

$$(H + r_{kk}I)y_k = c_k - \sum_{j=k+1}^n r_{kj}y_j.$$

Case 2. If $r_{k,k-1} \neq 0$, then equating columns $k-1$ and k in $HY + YR^T = \hat{C}$, it is easy to see that y_{k-1} and y_k are simultaneously computed by solving the $2m \times 2m$ linear system:

$$H(y_{k-1}, y_k) + (y_{k-1}, y_k) \begin{pmatrix} r_{k-1,k-1} & r_{k,k-1} \\ r_{k-1,k} & r_{kk} \end{pmatrix} (c_{k-1}, c_k) = \sum_{j=k+1}^n (r_{k-1,j}y_j, r_{k,j}y_j) = (d_{k-1}, d_k) \quad (8.5.7)$$

Note: The matrix of the system can be made upper triangular with two non zero subdiagonals, by reordering the variables suitably. The upper triangular system then can be solved using Gaussian elimination with partial pivoting.

Step 3. Recovery of the Original Solution. The solution X is recovered from Y as

$$X = VYU^T.$$

Algorithm 8.5.1 The Hessenberg-Schur Algorithm for $XA + BX = C$

Input: The matrices A , B , and C , respectively, of order $n \times n$, $m \times m$, and $m \times n$; $n \leq m$.

Output: The matrix X satisfying $XA + BX = C$

Step 1. Transform A^T to a real Schur matrix R , and B to an upper Hessenberg matrix H by orthogonal similarity:

$$U^T A^T U = R, \quad V^T B V = H$$

Form $\hat{C} = V^T C U$, and partition $\hat{C} = (c_1, \dots, c_n)$ by columns.

Step 2. Solve $HY + YR^T = \hat{C}$:

For $k = n, \dots, 1$ do until the columns of Y are computed

If $r_{k,k-1} = 0$, then compute y_k by solving the Hessenberg system:

$$(H + r_{kk}I)y_k = c_k - \sum_{j=k+1}^n r_{kj}y_j \quad (8.5.8)$$

Else, compute y_k and y_{k-1} by solving the system:

$$\begin{pmatrix} H + r_{k-1,k-1}I & r_{k-1,k}I \\ r_{k,k-1}I & H + r_{kk}I \end{pmatrix} \begin{pmatrix} y_{k-1} \\ y_k \end{pmatrix} = \begin{pmatrix} d_{k-1} \\ d_k \end{pmatrix}, \quad (8.5.9)$$

where

$$(d_{k-1}, d_k) = (c_{k-1}, c_k) - \sum_{j=k+1}^n (r_{k-1,j} y_j, r_{kj} y_j) \quad (8.5.10)$$

Step 3. Recover X : $X = VYU^T$.

Example 8.5.4 Consider solving the Sylvester equation using Algorithm 8.5.1 with

$$A = \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 2 & -1 \\ -3 & -2 & 2 \\ -2 & 1 & -1 \end{pmatrix} \text{ and } C = \begin{pmatrix} -6 & 9 \\ -2 & 2 \\ 2 & -5 \end{pmatrix}.$$

Step 1. Reduce A^T to upper real Schur form R and B to Hessenberg form H : $[U, R] = \text{schur}(A^T)$ gives

$$U = \begin{pmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{pmatrix}, \quad R = \begin{pmatrix} -2 & 0 \\ 0 & 2 \end{pmatrix}.$$

Similarly, $[V, H] = \text{hess}(B)$ gives

$$V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.8321 & -0.5547 \\ 0 & -0.5547 & 0.8321 \end{pmatrix}, \quad H = \begin{pmatrix} 0 & -1.1094 & -1.9415 \\ 3.6056 & -0.3071 & -1.5385 \\ 0 & -0.5385 & -2.6923 \end{pmatrix}.$$

Then

$$\hat{C} = V^T C U = \begin{pmatrix} -10.6066 & 2.1212 \\ -0.3922 & 1.1767 \\ 5.6874 & -1.7650 \end{pmatrix}$$

Step 2. Compute y_2 by solving the system: $(H + r_{22}I)y_2 = c_2$

$$y_2 = \begin{pmatrix} 4.2426 \\ -3.5301 \\ 5.2951 \end{pmatrix}$$

Compute y_1 by solving the system: $(H + r_{11}I)y_1 = c_1 - r_{21}y_2$

$$y_1 = \begin{pmatrix} 3.4345 \\ 6.8697 \\ -2.0004 \end{pmatrix}.$$

Thus $Y = (y_1, y_2)$ is

$$Y = \begin{pmatrix} 3.4345 & 4.2426 \\ 6.8697 & -3.5301 \\ -2.0004 & 5.2951 \end{pmatrix}.$$

Step 3. Recover X from Y :

$$X = VYU^T = \begin{pmatrix} 5.4286 & 0.5714 \\ -3.2571 & 3.2571 \\ 0.6286 & 8.3714 \end{pmatrix}.$$

Example 8.5.5 Consider solving $XA + BX = C$ using Algorithm 8.5.1 with

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \\ 7 & 8 & 9 & 1 \\ 10 & 0 & 0 & 0 \end{pmatrix}$$

and

$$C = \begin{pmatrix} 12 & 10 & 12 \\ 24 & 22 & 24 \\ 27 & 25 & 27 \\ 12 & 10 & 12 \end{pmatrix}.$$

Step 1. Reduce of A^T to real Schur form and B to Hessenberg form H :

$$\begin{aligned} U^T A^T U &= R = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \\ V^T B V &= H = \begin{pmatrix} 1 & -5.3766 & -0.3709 & -0.0886 \\ -12.8452 & 7.6545 & 5.3962 & -0.7695 \\ 0 & 10.6689 & 4.7871 & -0.2737 \\ 0 & 0 & -5.3340 & 1.5584 \end{pmatrix}. \\ U &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, V = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -3.114 & -0.7398 & -0.5965 \\ 0 & -5.449 & -0.3752 & 0.7498 \\ 0 & -0.7785 & 0.5585 & -0.2863 \end{pmatrix}, \end{aligned}$$

Compute

$$\hat{C} = \begin{pmatrix} 12 & -10 & 12 \\ -31.5292 & 28.2595 & -31.5292 \\ -21.1822 & 20.0693 & -21.1822 \\ 2.4949 & -2.7608 & 2.4949 \end{pmatrix}.$$

Step 2. Solution of the reduced problem: $HY + YR^T = \hat{C}$

Case 1. Since $r(3, 2)$ is 0, y_3 is obtained by solving: $(H + r_{33}I)y_3 = c_3$.

$$y_3 = (1, -1.6348, -0.5564, -0.1329)^T.$$

Case 2. Since $r_{21} \neq 0$, y_1 and y_2 are simultaneously computed by solving the system:

$$\begin{pmatrix} H + r_{11}I & r_{12}I \\ r_{21}I & H + r_{22}I \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix},$$

where $(d_1, d_2) = (c_1 - r_{13}y_3, c_2 - r_{23}y_3)$.

$$y_1 = \begin{pmatrix} 1 \\ -1.6348 \\ -0.5564 \\ -0.1329 \end{pmatrix}, \quad y_2 = \begin{pmatrix} -1 \\ 1.6348 \\ 0.5564 \\ 0.1329 \end{pmatrix}.$$

So,

$$Y = (y_1, y_2, y_3) = \begin{pmatrix} 1 & -1 & 1 \\ -1.6348 & 1.6348 & -1.6348 \\ -0.5564 & 0.5564 & -0.5564 \\ -0.1329 & 0.1329 & -0.1329 \end{pmatrix}$$

Step 3. $X = VYU^T = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

Flop-count:

1. Reduction to Hessenberg and real Schur forms: $\frac{10}{3}m^3 + 26n^3$.

2. Computation of \hat{C} : $2m^2n + 2mn^2$.

3. Computation of Y : $6m^2n + mn^2$.

(To obtain Y , it was assumed that S has $\frac{n}{2}(2 \times 2)$ bumps, which is the worst case.)

4. Computation of X : $2m^2n + 2mn^2$.

Total Flops: **Approximately** $(10m^3/3 + 26n^3 + 10m^2n + 5mn^2)$.

Numerical Stability of the Schur and Hessenberg-Schur Methods

The round-off error analysis of the Hessenberg-Schur algorithm for the Sylvester equation $XA + BX = C$ performed by Golub, Nash, and Van Loan (1979) shows that “the errors no worse in magnitude than $O(\|\phi^{-1}\| \epsilon)$ will contaminate the computed \hat{X} , where $\|\phi^{-1}\| = \frac{1}{\text{sep}(B, -A)}$, and ϵ is a small multiple of the machine precision μ .”

Specifically, if $\frac{\epsilon(2 + \epsilon)(\|A\|_2 + \|B\|_2)}{\text{sep}(B, -A)} < \frac{1}{2}$

then

$$\frac{\|X - \hat{X}\|_F}{\|X\|_F} \leq \frac{(9\epsilon + 2\epsilon^2)(\|A\|_F + \|B\|_F)}{\text{sep}(B, -A)}. \quad (8.5.11)$$

The above result shows that the quantity $\text{sep}(B, -A)$ will indeed influence the numerical accuracy of the computed solution obtained by the Hessenberg-Schur algorithm for the Sylvester equation (Note that $\text{sep}(B, -A)$ also appears in the perturbation bound (8.3.6)).

Similar remarks, of course, also hold for the Schur methods for the Lyapunov and Sylvester equations. We will have some more to say about the **backward error** of the computed solutions by these methods a little later in this Chapter.

MATCONTROL Notes: Algorithm 8.5.1 has been implemented in MATCONTROL function **sylvhrc**. The function **sylvhcsc** solves the Sylvester equation using **Hessenberg decomposition** of B and **complex-Schur decomposition** of A .

MATLAB NOTE: MATLAB function **lyap** in the form

$$X = \text{lyap} (A, B, C)$$

solves the Sylvester equation

$$AX + XB = -C$$

using **complex Schur decompositions** of both A and B .

8.5.4 The Schur Method for the Discrete Lyapunov Equation

We now briefly outline the Schur-method for the discrete Lyapunov equation:

$$A^T X A - X = C. \quad (8.5.12)$$

The method is due to Barraud (1977).

As before, we divide the process into three steps:

Step 1. Reduction of the problem.

Let $R = U^T A^T U$ be the upper real Schur form of the matrix A^T . Then the equation

$$A^T X A - X = C$$

reduces to

$$R Y R^T - Y = \hat{C}, \quad (8.5.13)$$

where $Y = U^T X U$ and $\hat{C} = U^T C U$.

Step 2. Solution of the Reduced Equation.

Let $R = (r_{ij})$, $Y = (y_1, y_2, \dots, y_n)$, and $\hat{C} = (c_1, c_2, \dots, c_n)$.

Consider two cases as before.

Case 1. $r_{k,k-1} = 0$, for some k .

In this case, y_k can be determined by solving the quasi-triangular system:

$$(r_{kk}R - I)y_k = c_k - R \sum_{j=k+1}^n r_{kj}y_j \quad (8.5.14)$$

In particular, if R is an upper triangular matrix, then y_n through y_1 can be computed successively by solving the triangular systems:

$$(r_{kk}R - I)y_k = c_k - R \sum_{j=k+1}^n r_{kj}y_j, \quad k = n, n-1, \dots, 2, 1. \quad (8.5.15)$$

Case 2. $r_{k,k-1} \neq 0$, for some k . In this case y_k and y_{k-1} can be simultaneously computed, as before.

For example, if $n = 3$, and $r_{2,1} \neq 0$, then y_2 and y_1 can be computed simultaneously by solving the system:

$$\begin{pmatrix} r_{11}R - I & r_{12}R \\ r_{21}R & r_{22}R - I \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} c_1 - r_{13}Ry_3, \\ c_2 - r_{23}Ry_3 \end{pmatrix}. \quad (8.5.16)$$

Step 3. Recovery of X from Y . Once Y is computed, X is recovered from Y as

$$X = UYU^T. \quad (8.5.17)$$

Example 8.5.6 Consider solving the discrete Lyapunov equation $A^T X A - X = C$ with

$$A = \begin{pmatrix} 0 & 2 & -1 \\ -3 & -2 & 2 \\ -2 & 1 & -1 \end{pmatrix} \text{ and } C = \begin{pmatrix} -2 & 2 & -3 \\ -8 & -6 & -5 \\ 11 & 13 & -2 \end{pmatrix}.$$

Step 1. Reduction to : $RYR^T - Y = \hat{C}$.

$$R = \begin{pmatrix} -1.3776 & 3.8328 & 1.3064 \\ -1.0470 & 0.8936 & -1.2166 \\ 0 & 0 & -2.5160 \end{pmatrix},$$

$$U = \begin{pmatrix} 0.7052 & 0.4905 & 0.5120 \\ 0.6628 & -0.7124 & -0.2304 \\ -0.2518 & -0.5019 & 0.8275 \end{pmatrix},$$

$$\hat{C} = \begin{pmatrix} -9.3174 & 1.9816 & -7.5863 \\ -2.1855 & -1.0425 & 2.4422 \\ 16.2351 & -3.4886 & 0.3600 \end{pmatrix}.$$

Step 2. Solution of the reduced equation: $RYR^T - Y = \hat{C}$.

Since $r_{32} = 0$, we compute y_3 by solving $(r_{33}R - I)y_3 = c_3$, which gives

$$y_3 = \begin{pmatrix} 2.6146 \\ 1.4323 \\ 0.0675 \end{pmatrix}.$$

Again, since $r_{21} \neq 0$, we simultaneously compute y_1, y_2 by solving:

$$\begin{pmatrix} r_{11}R - I & r_{12}R \\ r_{21}R & r_{22}R - I \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} c_1 - r_{13}Ry_3 \\ c_2 - r_{23}Ry_3 \end{pmatrix},$$

which gives

$$y_1 = \begin{pmatrix} 1.6987 \\ -1.0276 \\ -5.1219 \end{pmatrix},$$

and

$$y_2 = \begin{pmatrix} 2.6509 \\ 0.0703 \\ -3.0163 \end{pmatrix}.$$

Thus

$$Y = (y_1, y_2, y_3) = \begin{pmatrix} 1.6987 & 2.6509 & 2.6146 \\ -1.0276 & 0.0703 & 1.4323 \\ -5.1219 & -3.0163 & 0.0675 \end{pmatrix}.$$

Step 3. Recovery of X from Y :

$$\begin{aligned} X &= UYU^T \\ &= \begin{pmatrix} 0.1376 & -2.1290 & 2.4409 \\ 3.6774 & 0.1419 & -1.3935 \\ -5.1721 & -0.1678 & 1.5570 \end{pmatrix}. \end{aligned}$$

Verify: $A^T X A - X - C = O(10^{-13})$

Flop-Count. The Schur method for the discrete Lyapunov equation requires about $34n^3$ flops ($26n^3$ for the reduction of A to the real Schur form).

Round-off Properties. As in the case of the continuous-time Lyapunov equation, it can be shown [Exercise 29] that the computed solution \hat{X} of the discrete Lyapunov equation $A^T X A - X - C = 0$ satisfies the inequality

$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} \leq \frac{cm\mu}{\text{sep}_d(A^T, A)}, \quad (8.5.18)$$

where $m = \max(1, \|A\|_F^2)$ and c is a small constant.

Thus, the accuracy of the solution obtained by the Schur method for the discrete-Lyapunov equation depends upon the quantity $\text{sep}_d(A^T, A)$ (Note again that the $\text{sep}_d(A^T, A)$ appears in the perturbation bound (8.3.18)).

MATLAB Note: $X = \text{dlyap}(A, C)$ solves the discrete Lyapunov equation: $AXA^T - X = -C$, using **complex-Schur** decomposition of A .

MATCONTROL NOTES: MATCONTROL functions **lyaprsd** and **lyapcsd** solve the discrete-time Lyapunov equation using **real-Schur** and **complex-Schur** decomposition of A , respectively.

8.5.5 Residual and Backward Error in the Schur and Hessenberg-Schur Algorithms

We consider here the following questions: How small are the relative residuals obtained by the Schur and the Hessenberg-Schur algorithms? **Does a small relative residual guarantee that the solution is accurate?**

To answer these questions, we note that there are two major computational tasks with these algorithms:

First. The reduction of the matrices to the real Schur form and/or to the Hessenberg form.

Second. Solutions of certain linear systems.

We know that the reduction to the real Schur form of a matrix by the QR iteration method, and that to the Hessenberg form by Householder's or Givens' method, are backward stable (**See Chapter 4**).

And, if the linear systems are solved using Gaussian elimination with partial pivoting, followed by the technique of iterative refinement (which is the most practical way to solve a dense linear system), then it can be shown (Golub, Nash and Van Loan (1979), Higham (1996)) that the relative residual norm obtained by the **Hessenberg-Schur algorithm** for the Sylvester equation satisfies

$$\frac{\|C - (\hat{X}A + B\hat{X})\|_F}{\|\hat{X}\|_F} \leq c\mu(\|A\|_F + \|B\|_F), \quad (8.5.19)$$

where \hat{X} is the computed solution and c is a small constant depending upon m and n .

This means that the relative residual is guaranteed to be small. Note that this bound does not involve $\text{sep}(B, -A)$.

Does a small relative residual imply a small backward error? We will now consider this question.

To this end, let's recall that by **backward error** we mean the amount of perturbations to be made to the data so that an approximate solution is the exact solution to the perturbed problem. If the perturbations are small, then the algorithm is backward stable.

For the Sylvester equation problem, let's define (following Higham (1996)) the backward error of an approximate solution Y of the Sylvester equation $XA + BX = C$ by

$$\begin{aligned}\nu(Y) &= \min \{ \varepsilon : Y(A + \Delta A) + (B + \Delta B)Y = C + \Delta C, \\ &\quad \|\Delta A\|_F \leq \varepsilon\alpha, \|\Delta B\|_F \leq \varepsilon\beta, \|\Delta C\|_F \leq \varepsilon\gamma \},\end{aligned}$$

where α, β and γ are tolerances. The most common choice is

$$\alpha = \|A\|_F, \beta = \|B\|_F, \gamma = \|C\|_F.$$

This choice yields the **normwise relative backward error**.

As earlier, we assume that A is $n \times n$ and B is $m \times m$, and $m \geq n$.

It has then been shown by Higham (1996) that

$$\nu(Y) \leq \delta \frac{\|Res(Y)\|_F}{(\alpha + \beta)\|Y\|_F + \gamma}, \quad (8.5.20)$$

where $Res(Y) = C - (YA + BY)$ is the residual and

$$\delta = \frac{(\alpha + \beta)\|Y\|_F + \gamma}{\sqrt{(\alpha^2\sigma_m^2 + \beta^2\sigma_n^2 + \gamma^2)}}. \quad (8.5.21)$$

Here $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ are the singular values of Y , and $\sigma_{n+1} = \dots = \sigma_m = 0$.

The special case when $m = n$ is interesting. In this case

$$\delta = \frac{(\|A\|_F + \|B\|_F)\|Y\|_F + \|C\|_F}{((\|A\|_F^2 + \|B\|_F^2)\sigma_{\min}^2(Y) + \|C\|_F^2)^{\frac{1}{2}}} \quad (8.5.22)$$

Thus δ is large only when

$$\|Y\|_F \gg \sigma_{\min}(Y) \quad \text{and} \quad \|Y\|_F \gg \frac{\|C\|_F}{\|A\|_F + \|B\|_F} \quad (8.5.23)$$

In other words, **δ is large when Y is ill-conditioned and $\|Y\|_F$ is large**.

In the general case ($m \neq n$), δ can also be large if $\|B\|$ is large compared to the rest of the data. In these cases the Sylvester equation is badly scaled.

Also, **note that if only A and B are perturbed, then δ is large whenever Y is ill-conditioned**.

This is because in this case,

$$\delta \geq \|Y\|_F \|Y^\dagger\|_2 \approx \text{Cond}_2(Y)$$

(for any m and n); so, δ is large whenever Y is ill-conditioned.

From above discussions, we see that “**the backward error of an approximate solution to the Sylvester equation can be arbitrarily larger than its relative residual**” (Higham (1996)). The same remark, of course, also holds for the Lyapunov equation, as we will see below.

Backward Error for the Lyapunov Equation

Since in the Lyapunov equation, $B = A^T$, and thus $\beta = \alpha$, we have the following bound for the backward error for the Lyapunov equation.

Let Y be an approximate solution of the Lyapunov equation $XA + A^T X = C$, and let $\nu(Y)$ denote the backward error:

$$\nu(Y) = \min\{\epsilon : Y(A+\Delta A)+(A+\Delta A)^T Y = C+\Delta C, \|\Delta A\| \leq \epsilon\alpha, \Delta C = (\Delta C)^T, \|\Delta C\|_F \leq \epsilon\gamma\}.$$

Then

$$\nu(Y) \leq \delta \frac{\|Res(Y)\|_F}{2\alpha \|Y\|_F + \gamma}, \quad (8.5.24)$$

where

$$\delta = \frac{\sqrt{2}(2\alpha \|Y\|_F + \gamma)}{\sqrt{4\alpha^2\lambda_*^2 + \gamma^2}}, \quad \alpha = \|A\|_F, \quad \gamma = \|C\|_F, \quad (8.5.25)$$

and $Res(Y) = C - YA - A^T Y$ and $\lambda_* = \min_i |\lambda_i|$; λ_i 's are the eigenvalues of A .

8.5.6 A Hessenberg Method for the Sylvester Equation: $AX + XB = C$

Though the Schur methods and the Hessenberg-Schur methods are numerically effective for the Lyapunov and the Sylvester equations and are widely used in practice, it would be, however nice to have methods that would require reduction of the matrices A and B to Hessenberg forms only. Note that the reduction to a Hessenberg form is preliminary to that of the Real Schur form. Thus such Hessenberg methods, will be more efficient than the Hessenberg-Schur method. We show below how such a Hessenberg method for the Sylvester equation can be developed. The method is an extension of a Hessenberg method for the Lyapunov equation by Datta and Datta (1976), and is an efficient implementation of an idea of Kreisselmeier (1972). *It answers affirmatively a question raised by Charles Van Loan (1982) as to if a method can be developed to solve the Lyapunov equation just by transforming A to a Hessenberg matrix.*

Step 1. Reduction of the Problem to a Hessenberg Problem

Transform A to a lower Hessenberg matrix H_1 , and B to another lower Hessenberg matrix H_2 :

$$\begin{aligned} U^T AU &= H_1, \\ V^T BV &= H_2. \end{aligned}$$

(Assume that both H_1 and H_2 are **unreduced**).

Then $AX + XB = C$ becomes

$$H_1 Y + Y H_2 = C' \text{ where } Y = U^T X V, \quad C' = U^T C V.$$

Step 2. Solution of the Reduced Problem: $H_1 Y + Y H_2 = C'$

Let $Y = (y_1, y_2, \dots, y_n)$ and $H_2 = (h_{ij})$.

Then the equation $H_1Y + YH_2 = C'$ is equivalent to

$$\begin{aligned} H_1y_n + h_{n-1,n}y_{n-1} + h_{nn}y_n &= c'_n \\ H_1y_{n-1} + h_{n-2,n-1}y_{n-2} + h_{n-1,n-1}y_{n-1} + h_{nn-1}y_n &= c'_{n-1} \\ &\vdots \\ H_1y_1 + h_{11}y_1 + h_{21}y_2 + \cdots + h_{n1}y_n &= c'_1 \end{aligned}$$

Eliminating y_1 through y_{n-1} , we have

$$Ry_n = d$$

where $R = \frac{1}{\prod_{i=2}^n h_{i-1,i}} \phi(-H_1)$; $\phi(x)$, being the characteristic polynomial of H_1 and the vector d is defined in **Step 4** below.

Thus, once y_n is obtained by solving the system $Ry_n = d$, y_{n-1} through y_1 can be computed recursively:

$$y_{i-1} = -\frac{1}{h_{i-1,i}} \left(H_1y_i + \sum_{j=i}^n h_{ji}y_j - c'_i \right), \quad i = n, n-1, \dots, 2.$$

Step 3. Computing the matrix R of Step 2.

It is well-known (see Datta and Datta (1976)) that by knowing only one row or a column of a polynomial matrix in an unreduced Hessenberg matrix, the other rows or columns of the matrix polynomial can be generated recursively.

Realizing that the matrix R is basically a polynomial matrix in the lower Hessenberg matrix H_1 , its computation is greatly facilitated.

Thus if $R = (r_1, \dots, r_n)$, then, knowing r_n ; r_{n-1} through r_1 can be generated recursively as follows:

$$r_{k-1} = \frac{1}{h'_{k-1,k}} \left(H_1r_k - \sum_{i=k}^n h'_{ik}r_i \right), \text{ where } H_1 = (h'_{ij}); k = n, n-1, \dots, 2$$

It therefore remains to know how to compute r_n . This can be done as follows.

Set $\theta_n = e_n = (0, 0, 0, \dots, 0, 1)^T$ and then compute θ_{n-1} through θ_0 recursively by using

$$\theta_{i-1} = -\frac{1}{h_{i-1,i}} \left(H_1\theta_i + \sum_{j=i}^n h_{ji}\theta_j \right), \quad i = n, n-1, \dots, 1,$$

then, it can be shown (Datta and Datta (1976)) that

$$r_n = \theta_0, \text{ setting } h_{01} = 1.$$

Step 4. Computing the vector d of Step 2

The vector d can also be generated from the above recursion. Thus, starting with $z_n = 0$ (a zero vector), if z_{n-1} through z_0 are generated recursively using

$$z_{i-1} = -\frac{1}{h_{i-1,i}} \left(H_1 z_i + \sum_{j=i}^n h_{ji} z_j - c'_i \right), i = n, \dots, 2, 1.$$

then $d = -z_0$.

Step 5. Recovery of the original solution X from Y :

$$X = U Y V^T.$$

Remarks: It is to be noted that the method, as presented above, is of theoretical interest only at present. There are possible numerical difficulties. For example, if one or more of the entries of the subdiagonal of the Hessenberg matrix H_2 are small, a large round-off error can be expected in computing y_{i-1} in Step 2. A detailed study on the numerical behavior of the method is necessary, before recommending it for practical use. Probably, some modification will be necessary to make it a working numerical algorithm. The reason for including this method here is to show that *a method for the Sylvester equation can be developed just by passing through the Hessenberg transformations of the matrices A and B*.

Example 8.5.7 Consider solving the Sylvester equation $AX + XB = C$ with the following data

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \\ 7 & 8 & 9 & 1 \\ 10 & 0 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad C = \begin{pmatrix} 12 & 10 & 12 \\ 24 & 22 & 24 \\ 27 & 25 & 27 \\ 12 & 10 & 12 \end{pmatrix}$$

Step 1: Reduction of A and B to Hessenberg forms:

$$H_1 = \begin{pmatrix} 1.0000 & -5.3852 & 0 & 0 \\ -12.8130 & 8.7241 & 5.1151 & 0 \\ 0.8337 & 10.3127 & 4.6391 & 0.1586 \\ 0.3640 & 1.3595 & -4.8552 & 0.6368 \end{pmatrix}, \quad U = \begin{pmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & -0.3714 & -0.6009 & -0.7078 \\ 0 & -0.5571 & -0.4657 & 0.6876 \\ 0 & -0.7428 & 0.6497 & -0.1618 \end{pmatrix}$$

$$H_2 = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad C' = \begin{pmatrix} 12.0000 & -10.0000 & 12.0000 \\ -32.8681 & 29.5256 & -32.8681 \\ -19.1978 & 18.3641 & -19.1978 \\ -0.3640 & -0.0000 & -0.3640 \end{pmatrix}$$

Step 2: Solution of the reduced problem:

Since the matrix H_2 is **reduced** ($h_{23} = 0$), instead of an algorithm breakdown, the set of equations for y_1, y_2, y_3 decouple and we obtain:

$$\begin{aligned} H_1 y_3 + h_{33} y_3 &= c'_3 \\ H_1 y_2 + h_{12} y_1 + h_{22} y_2 &= c'_2 - h_{32} y_3 = \hat{c}_2 \\ H_1 y_1 + h_{11} y_1 + h_{21} y_2 &= c'_1 - h_{31} y_3 = \hat{c}_1 \end{aligned}$$

The vector y_3 is obtained as the solution of the first system, and once y_3 is known, \hat{c}_2 and \hat{c}_3 can be easily computed.

$$y_3 = \begin{pmatrix} 1.0000 \\ -1.6713 \\ -0.4169 \\ -0.1820 \end{pmatrix}, \quad \hat{c}_2 = \begin{pmatrix} -10.0000 \\ 29.5256 \\ 18.3641 \\ -0.0000 \end{pmatrix}, \quad \hat{c}_1 = \begin{pmatrix} 12.0000 \\ -32.8681 \\ -19.1978 \\ -0.3640 \end{pmatrix}$$

We now proceed to compute y_2 and y_1 as follows:

Step 4: Computation of the vector d : starting from $z_2 = (0 \ 0 \ 0 \ 0)^T$,

$$z_1 = -\frac{1}{h_{12}}(H_1 z_2 + h_{22} z_2 - \hat{c}_2) = (-10.0000 \ 29.5256 \ 18.3641 \ -0.0000)^T$$

$$d = -z_0 = \frac{1}{1}(H_1 z_1 + h_{11} z_1 + h_{21} z_2 - \hat{c}_1) = (-191.0000 \ 542.0447 \ 418.9050 \ -52.2985)^T$$

Step 3: Computation of the matrix R . Starting from $\theta_2 = (0 \ 0 \ 0 \ 1)^T$,

$$\theta_1 = -\frac{1}{h_{12}}(H_1 \theta_2 + h_{22} \theta_2) = (0 \ 0 \ -0.1586 \ -1.6368)^T$$

$$\theta_0 = -\frac{1}{1}(H_1 \theta_1 + h_{11} \theta_1 + h_{21} \theta_2) = (0 \ 0.8112 \ 1.1538 \ 2.9092)^T$$

and now, starting from $r_4 = \theta_0$, we obtain

$$r_3 = \frac{1}{h'_{34}}(H_1 r_4 - h'_{44} r_4) = (-27.5462 \ 78.5858 \ 84.7805 \ -28.3717)^T$$

$$r_2 = \frac{1}{h'_{23}}(H_1 r_3 - h'_{33} r_3 - h'_{43} r_4) = (-63.1364 \ 217.3103 \ 154.1618 \ -36.5856)^T$$

$$r_1 = \frac{1}{h'_{12}}(H_1 r_2 - h'_{22} r_2 - h'_{32} r_3 - h'_{42} r_4) = (74.0000 \ -145.9565 \ -125.7098 \ -20.1428)^T$$

which gives

$$R = \begin{pmatrix} 74.0000 & -63.1364 & -27.5462 & 0 \\ -145.9565 & 217.3103 & 78.5858 & 0.8112 \\ -125.7098 & 154.1618 & 84.7805 & 1.1538 \\ -20.1428 & -36.5856 & -28.3717 & 2.9092 \end{pmatrix}$$

and now $Ry_2 = d$ gives

$$y_2 = (-1.0000 \ 1.6713 \ 0.4169 \ 0.1820)^T$$

and finally we compute

$$y_1 = (1.0000 \ -1.6713 \ -0.4169 \ -0.1820)^T.$$

Therefore, the solution of the reduced problem is

$$Y = \begin{pmatrix} 1.0000 & -1.0000 & 1.0000 \\ -1.6713 & 1.6713 & -1.6713 \\ -0.4169 & 0.4169 & -0.4169 \\ -0.1820 & 0.1820 & -0.1820 \end{pmatrix}.$$

The original solution X is then recovered via $X = UYV^T$

$$X = \begin{pmatrix} 1.0000 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 1.0000 \\ 1.0000 & 1.0000 & 1.0000 \end{pmatrix}.$$

Verification: $\|AX + XB - C\|_2 = 5.6169 \times 10^{-14}$.

MATCONTROL NOTE: The Hessenberg methods for the Sylvester and Lyapunov equations have been implemented in MATCONTROL functions **sylvhess** and **lyaphess**, respectively.

8.5.7 The Hessenberg-Schur Method for the Discrete Sylvester Equation

In some applications, one needs to solve a general discrete Sylvester equation:

$$BXA + C = X.$$

The Schur method for the discrete Lyapunov equation described in [Section 8.5.4](#) can be easily extended to solve this equation.

Assume that the order of A is smaller than that of B . $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{m \times m}$.

Let the matrices A^T and B be transformed, respectively, to an upper real Schur matrix R and an upper Hessenberg matrix H by orthogonal similarity:

$$\begin{aligned} U^T A^T U &= R, \\ V^T B V &= H. \end{aligned}$$

Then

$$BXA + C = X \quad \text{becomes} \quad HYR^T + \hat{C} = Y,$$

where $Y = V^T X U$, $\hat{C} = V^T C U$. Let $Y = (y_1, \dots, y_n)$, and $\hat{C} = (c_1, c_2, \dots, c_n)$.

The reduced equation can now be solved in exactly the same way as in the Hessenberg-Schur algorithm for the Sylvester equation ([Algorithm 8.5.1](#)). This is left as an exercise [[Exercise 30](#)] for the readers.

MATCONTROL Note. MATCONTROL function **sylvhcsd** solves the discrete-time Sylvester equation, based on **complex-Schur** decomposition of A .

8.6 Direct Computations of the Cholesky Factors of Symmetric Positive Definite Solutions of Lyapunov Equations

In this section we describe methods for finding the Cholesky factors of the symmetric positive definite solutions of both continuous-time and discrete-time Lyapunov equations, without explicitly computing such solutions.

8.6.1 Computing the Cholesky Factor of the Positive Definite Solution of the Lyapunov Equation

Consider first the Lyapunov equation

$$XA + A^T X = -C^T C \quad (8.6.1)$$

where A is an $n \times n$ stable matrix (i.e., all the eigenvalues $\lambda_1, \dots, \lambda_n$ have negative real parts), and C is an $r \times n$ matrix.

The above equation admits a unique symmetric positive semidefinite solution X . Thus, such a solution matrix X has the Cholesky factorization $X = Y^T Y$ where Y is upper triangular.

In several applications, all that is needed is the matrix Y ; X is not needed as such. One such application is **model reduction problem via internal balancing and the Schur method for model reduction (Chapter 14)**, where the Cholesky factors of the controllability and observability Grammians are needed.

In these applications, it might be computationally more attractive to obtain the matrix Y directly without solving the equation for X , because X can be considerably more ill-conditioned than Y , because, $\text{Cond}_2(X) = (\text{Cond}_2(Y))^2$. Also, it may not be computationally desirable to form the right hand side matrix $-C^T C$ explicitly; there may be a significant loss of accuracy in this explicit formation.

We describe below a procedure due to Hammarling (1982) for finding the Cholesky factor Y without explicitly computing X and without forming the matrix product $C^T C$.

Reduction of the problem

Substituting $X = Y^T Y$ in the equation (8.6.1), we have

$$(Y^T Y)A + A^T(Y^T Y) = -C^T C. \quad (8.6.2)$$

The challenge is now to compute Y without explicitly forming the product $C^T C$.

Let $S = U^T A U$, where S is in upper real Schur form and U is orthogonal. Let

$$CU = QR$$

be the QR factorization of CU .

Then the equation (8.6.2) becomes

$$S^T \left(\hat{Y}^T \hat{Y} \right) + \left(\hat{Y}^T \hat{Y} \right) S = -R^T R \quad (8.6.3)$$

where $\hat{Y} = YU$ and $R^T R = (CU)^T CU$.

Solution of the Reduced Equation

To obtain \hat{Y} from (8.6.3) without explicitly forming $R^T R$, we partition \hat{Y} , R and S as follows:

$$\hat{Y} = \begin{pmatrix} \hat{y}_{11} & \hat{y}^T \\ 0 & Y_1 \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r^T \\ 0 & R_1 \end{pmatrix}, \quad S = \begin{pmatrix} s_{11} & s^T \\ 0 & S_1 \end{pmatrix} \quad (8.6.4)$$

where s_{11} is a scalar (a real eigenvalue in the real Schur form S) or a 2×2 matrix (“Schur bump”, corresponding to a pair of complex conjugate eigenvalues in the matrix S); and \hat{y} , r and s are either column vectors or matrices with two columns.

Since \hat{Y} satisfies (8.6.3) we can show, after some algebraic manipulations, that \hat{y}_{11} , \hat{y} and Y_1 satisfy the following equations:

$$s_{11}^T (\hat{y}_{11}^T \hat{y}_{11}) + (\hat{y}_{11}^T \hat{y}_{11}) s_{11} = -r_{11}^T r_{11} \quad (8.6.5)$$

$$S_1^T \hat{y} + \hat{y} (\hat{y}_{11} s_{11} \hat{y}_{11}^{-1}) = -r\alpha - s\hat{y}_{11}^T \quad (8.6.6)$$

$$S_1^T (Y_1^T Y_1) + (Y_1^T Y_1) S_1 = -\hat{R}_1^T \hat{R}_1 \quad (8.6.7)$$

where $\alpha = r_{11} \hat{y}_{11}^{-1}$, $\hat{R}_1^T \hat{R}_1 = R_1^T R_1 + uu^T$ and $u = r - \hat{y}\alpha^T$.

Since $R_1^T R_1$ is positive definite, so is $\hat{R}_1^T \hat{R}_1$.

Note that the matrix \hat{R}_1 can be easily computed, once R_1 and u are known, from the QR factorization:

$$\begin{pmatrix} u^T \\ R_1 \end{pmatrix} = \hat{Q} \hat{R}_1. \quad (8.6.8)$$

The equations (8.6.7) is of the same form as the original reduced equation (8.6.2), **but is of smaller order. This is the key observation.**

The matrices S_1 , Y_1 , and \hat{R}_1 can now be partitioned further as in (8.6.4), and the whole process can be repeated. The process is continued until \hat{Y} is completely determined.

Recovery of the solution

Once \hat{Y} is obtained, the “ R -matrix” \tilde{Y} of the QR factorization $\tilde{Q}\tilde{Y} = \hat{Y}U^T$ will be an upper triangular matrix that will satisfy the equation (8.6.7).

Since Y is required to have positive diagonal entries, we will take

$$Y = \text{diag}(\text{sign}(\tilde{y}_{11}), \dots, \text{sign}(\tilde{y}_{nn})) \tilde{Y}.$$

Algorithm 8.6.1 Algorithm for the Direct Cholesky Factor of the Symmetric Positive Definite Solution of the Lyapunov Equation.

Inputs: A - An $n \times n$ matrix

C - An $r \times n$ matrix.

Output: Y - The Cholesky factor of the symmetric positive definite solution of the Lyapunov equation: $XA + A^T X = -C^T C$.

Assumption. A is stable.

Step 1. Find the real Schur form S of A : $U^T A U = S$.

Step 2. Find the QR factorization of the $r \times n$ matrix CU : $CU = QR$.

Step 3. Partition $R = \begin{pmatrix} r_{11} & r^T \\ 0 & R_1 \end{pmatrix}$, $S = \begin{pmatrix} s_{11} & s^T \\ 0 & S_1 \end{pmatrix}$.

Step 4. Find $\hat{Y} = \begin{pmatrix} \hat{y}_{11} & \hat{y}^T \\ 0 & Y_1 \end{pmatrix}$ as follows:

4.1 Compute \hat{y}_{11} from $s_{11}^T(\hat{y}_{11}^T \hat{y}_{11}) + (\hat{y}_{11}^T \hat{y}_{11})s_{11} = -r_{11}^T r_{11}$

4.2 Compute $\alpha = r_{11} \hat{y}_1^{-1}$

4.3 Solve for \hat{y} : $S_1^T \hat{y} + \hat{y}(\hat{y}_{11} s_{11} \hat{y}_{11}^{-1}) = -r\alpha - s\hat{y}_{11}^T$

4.4 Compute $u = r - \hat{y}\alpha^T$ and then find the QR factorization of $\begin{pmatrix} u^T \\ R_1 \end{pmatrix}$:

$$Q\hat{R}_1 = \begin{pmatrix} u^T \\ R_1 \end{pmatrix}.$$

Step 5. Set $S = S_1$, $R = \hat{R}_1$ and return to Step 3 and continue until \hat{Y} is completely determined.

Step 6. Compute \tilde{Y} from the QR factorization of $\hat{Y}U^T$: $\hat{Y}U^T = Q\tilde{Y}$. Let $\tilde{Y} = (\tilde{y}_{ij})$.

Step 7. Compute $Y = \begin{pmatrix} sign(\tilde{y}_{11}) & & 0 \\ & \ddots & \\ 0 & & sign(\tilde{y}_{nn}) \end{pmatrix} \tilde{Y}$.

Example 8.6.1 Consider solving the equation (8.6.1) for the Cholesky factor Y with

$$A = \begin{pmatrix} -0.9501 & 0.5996 & 0.2917 \\ 0.6964 & -1.0899 & -0.6864 \\ 0 & 0.0571 & -6.6228 \end{pmatrix}, \quad C = (1, 1, 1).$$

Step 1. Reduction of A to real Schur form: $[U, S] = \text{schur}(A)$ gives

$$U = \begin{pmatrix} -0.7211 & -0.6929 & 0.0013 \\ -0.6928 & 0.7210 & -0.0105 \\ -0.0063 & 0.0084 & 0.9999 \end{pmatrix}$$

$$S = \begin{pmatrix} -0.3714 & 0.0947 & 0.3040 \\ 0 & -1.6762 & -0.7388 \\ 0 & 0 & -6.6152 \end{pmatrix}.$$

Step 2. The QR factorization of $CU : [Q, R] = \mathbf{qr}(CU)$ gives

$$R = (-1.4202 \quad 0.0366 \quad 0.9908).$$

Step 3.

$$\begin{aligned} r_{11} &= -1.4202, & s_{11} &= -0.3714 \\ r &= \begin{pmatrix} 0.0366 \\ 0.9908 \end{pmatrix}, & S_1 &= \begin{pmatrix} -1.6962 & -0.7388 \\ 0 & -6.6152 \end{pmatrix}, \\ s &= \begin{pmatrix} 0.0947 \\ 0.3040 \end{pmatrix}. \end{aligned}$$

Step 4. Compute \hat{y}_{11}

$$\hat{y}_{11} = 1.6479.$$

$$\alpha = r_{11}\hat{y}_{11}^{-1} = -0.8619$$

Solve for \hat{y} :

$$(S_1^T + \hat{y}_{11}s_{11}\hat{y}_{11}^{-1}I)\hat{y} = -r\alpha - s\hat{y}_{11}^T$$

or

$$\begin{pmatrix} -2.0476 & 0 \\ -0.7388 & -6.9866 \end{pmatrix}\hat{y} = \begin{pmatrix} -0.1245 \\ 0.3530 \end{pmatrix}$$

$$\hat{y} = \begin{pmatrix} 0.0608 \\ -0.0569 \end{pmatrix}.$$

$$u = r - \hat{y}\alpha^T = \begin{pmatrix} 0.0890 \\ 0.9418 \end{pmatrix}.$$

$$R_1 = 0$$

$$\hat{R}_1 = (0.0890, 0.9418)$$

Step 5. Solution of the reduced 2×2 problem:

$$S \equiv S_1 = \begin{pmatrix} -1.6762 & -0.7388 \\ 0 & -6.6152 \end{pmatrix}$$

$$R \equiv \hat{R}_1 = (0.0890, 0.9418)$$

$$\hat{y}_{11} = 0.0486, \hat{y} = (0.2036), \hat{R}_1 = 0.5689.$$

Solution of the final 1×1 problem

$$\begin{aligned} S &= -6.6152 \\ R &= 0.5689 \\ \hat{y}_{11} &= 0.1564 \end{aligned}$$

Thus

$$\hat{Y} = \begin{pmatrix} 1.6479 & 0.0608 & -0.0569 \\ 0 & 0.0486 & 0.02036 \\ 0 & 0 & 0.1564 \end{pmatrix}$$

Step 6. Compute $\tilde{Y} : [Q_1, \tilde{Y}] = \mathbf{qr}(\hat{Y}U^T)$

$$\tilde{Y}_1 = \begin{pmatrix} 1.2309 & 1.0960 & 0.0613 \\ 0 & -0.0627 & -0.2011 \\ 0 & 0 & 0.1623 \end{pmatrix}.$$

$$\text{Step 7. } Y = \begin{pmatrix} 1.2309 & 1.0960 & 0.0613 \\ 0 & 0.0627 & 0.2011 \\ 0 & 0 & 0.1623 \end{pmatrix}.$$

MATCONTROL NOTE: Algorithm 8.6.1 has been implemented in MATCONTROL functions `lyapchlc`.

Remarks: Note that it is possible to arrange the computation of Y with a different form of partitioning than as shown in (8.6.4). For example, let us partition matrices \hat{Y} , R and S as follows

$$\hat{Y} = \begin{pmatrix} Y_{11} & y \\ 0 & y_1 \end{pmatrix}, \quad R = \begin{pmatrix} R_{11} & r \\ 0 & r_1 \end{pmatrix}, \quad S = \begin{pmatrix} S_{11} & s \\ 0 & s_1 \end{pmatrix} \quad (8.6.9)$$

where y_1 , r_1 and s_1 are scalars or 2×2 matrices and y , r and s are either column vectors or matrices with two columns.

Then, similar to equations (8.6.5), (8.6.6) and (8.6.7), one will obtain three equations. For example, the first one will be just the deflated version of the original equation.

$$S_{11}^T (Y_{11}^T Y_{11}) + (Y_{11}^T Y_{11}) S_{11} = -R_{11}^T R_{11}. \quad (8.6.10)$$

Suppose that the solution Y_{11} of this deflated equation has been computed, then the second and third equations will give us the expressions for y and y_1 .

By using this new partitioning, the original algorithm of Hammarling (1982) can be slightly improved.

In the following we will use this partitioning to solve the discrete equation.

8.6.2 Computing the Cholesky Factor of the Positive Definite Solution of the Discrete Lyapunov Equation

Consider now the discrete Lyapunov equation:

$$A^T X A + C^T C = X \quad (8.6.11)$$

where A is an $n \times n$ discrete-stable matrix (i.e., all the eigenvalues $\lambda_1, \dots, \lambda_n$ are inside unit circle) and C is an $r \times n$ matrix.

Then the equation (8.6.11) admits a unique symmetric positive semidefinite solution X . Such a solution matrix X has the Cholesky factorization: $X = Y^T Y$, where Y is upper triangular. We would obtain the matrix Y directly without solving the equation (8.6.11) for X . Substituting $X = Y^T Y$ into the equation (8.6.11), we have

$$A^T (Y^T Y) A + C^T C = Y^T Y \quad (8.6.12)$$

As in the case of the continuous-time Lyapunov equation (8.6.1), we now outline a method for finding Y of (8.6.12) without computing X and without forming the matrix $C^T C$.

Reduction of the problem.

Let $S = U^T A U$, where S is in upper real Schur form and U is an orthogonal matrix. Let

$$Q_1 R = C U$$

be the economy QR factorization of the matrix $C U$.

Then the equation (8.6.12) becomes

$$S^T (\hat{Y}^T \hat{Y}) S + R^T R = \hat{Y}^T \hat{Y} \quad (8.6.13)$$

where $\hat{Y} = Y U$ and $R^T R = (C U)^T C U$.

Solution of the Reduced Equation

To obtain \hat{Y} from (8.6.13) without forming $R^T R$ explicitly, we partition \hat{Y} , R and S as

$$\hat{Y} = \begin{pmatrix} Y_{11} & y \\ 0 & y_1 \end{pmatrix}, \quad R = \begin{pmatrix} R_{11} & r \\ 0 & r_1 \end{pmatrix}, \quad S = \begin{pmatrix} S_{11} & s \\ 0 & s_1 \end{pmatrix}$$

From (8.6.13), we then see that Y_{11} , y and y_1 satisfy the following equations:

$$S_{11}^T (Y_{11}^T Y_{11}) S_{11} + R_{11}^T R_{11} = (Y_{11}^T Y_{11}) \quad (8.6.14)$$

$$Y_{11}^T y - (Y_{11} S_{11})^T y s_1 = S_{11}^T Y_{11}^T Y_{11} s + R_{11}^T r \quad (8.6.15)$$

$$s_1^T y_1^T y_1 s_1 + (r_1^T r_1 + r^T r + (Y_{11} s + y s_1)^T (Y_{11} s + y s_1) - y^T y) = y_1^T y_1 \quad (8.6.16)$$

The equation (8.6.14) is of the same form as the original reduced equation (8.6.12), **but is of smaller order.**

Suppose that we have already computed the solution Y_{11} of this equation. Then y can be obtained from (8.6.15) by solving a linear system and, finally, (8.6.16) gives us y_1 .

Recovery of the solution.

Once \hat{Y} is obtained, the “R-matrix” \tilde{Y} of the QR factorization of the matrix $\hat{Y} U^T : \tilde{Q} \tilde{Y} = \hat{Y} U^T$ will be the upper triangular matrix that will solve the equation (8.6.12).

Since Y has to have positive diagonal entries, we take

$$Y = \text{diag}(\text{sign}(\tilde{y}_{11}), \dots, \text{sign}(\tilde{y}_{nn})) \tilde{Y}.$$

Algorithm 8.6.2 Algorithm for the Direct Cholesky Factor of the Symmetric Positive Definite Solution of the Discrete Lyapunov Equation

Inputs: A - An $n \times n$ matrix

C - An $r \times n$ matrix

Output: Y - The Cholesky factor Y of the symmetric positive definite solution X of the discrete Lyapunov Equation: $A^T X A + C^T C = X$

Assumption: A is discrete-stable; that is all its eigenvalues have modulii less than 1.

Step 1. Find the Real Schur form S of A : $U^T A U = S$

Step 2. Find the (economy size) QR factorization of the $r \times n$ matrix $C U$: $Q R = C U$

Step 3. Partition $S = \begin{pmatrix} S_{11} & * \\ 0 & * \end{pmatrix}$, $R = \begin{pmatrix} R_{11} & * \\ 0 & * \end{pmatrix}$, $Y = \begin{pmatrix} Y_{11} & * \\ 0 & * \end{pmatrix}$, where S_{11} is a scalar or 2×2 matrix (Schur bump).

Compute Y_{11} from $S_{11}^T (Y_{11}^T Y_{11}) S_{11} + R_{11}^T R_{11} = Y_{11}^T Y_{11}$.

Step 4. Do while dimension of $Y_{11} <$ dimension of S

4.1. Partition $Y = \begin{pmatrix} Y_{11} & y & * \\ 0 & y_1 & * \\ 0 & 0 & * \end{pmatrix}$, $S = \begin{pmatrix} S_{11} & s & * \\ 0 & s_1 & * \\ 0 & 0 & * \end{pmatrix}$, $R = \begin{pmatrix} R_{11} & r & * \\ 0 & r_1 & * \\ 0 & 0 & * \end{pmatrix}$, where s_1 is 1×1 scalar or 2×2 Schur bump.

4.2. Compute y from $Y_{11}^T y - (Y_{11} S_{11})^T y s_1 = S_{11}^T Y_{11}^T Y_{11} s + R_{11}^T r$

4.3. Compute y_1 from

$$s_1^T y_1^T y_1 s_1 + (r_1^T r_1 + r^T r + (Y_{11} s + y s_1)^T (Y_{11} s + y s_1) - y^T y) = y_1^T y_1$$

4.4. Go to Step 4 with $Y_{11} \equiv \begin{pmatrix} Y_{11} & y \\ 0 & y_1 \end{pmatrix}$

Step 5. Compute \tilde{Y} from the QR factorization of $Y_{11}U^T : Q\tilde{Y} = Y_{11}U^T$.

Let $\tilde{Y} = (\tilde{y}_{ij})$.

Step 6. Compute $Y = \begin{pmatrix} sign(\tilde{y}_{11}) & & 0 \\ & \ddots & \\ 0 & & sign(\tilde{y}_{nn}) \end{pmatrix} \tilde{Y}$.

Example 8.6.2 Consider solving the equation (8.6.12) for the Cholesky factor Y with

$$A = \begin{pmatrix} -0.1973 & -0.0382 & 0.0675 \\ -0.1790 & -0.3042 & -0.0544 \\ 0.0794 & 0.0890 & -0.1488 \end{pmatrix}$$

and

$$C = \begin{pmatrix} 0.0651 & 0.1499 & 0.2917 \\ 0.1917 & 0.0132 & 0.4051 \end{pmatrix}$$

Step 1. Reduction of A to the real Schur form: $[U, S] = \text{schur}(A)$ gives

$$U = \begin{pmatrix} -0.3864 & -0.7790 & 0.4938 \\ -0.7877 & 0.5572 & 0.2627 \\ 0.4798 & 0.2875 & 0.8290 \end{pmatrix}$$

$$S = \begin{pmatrix} -0.3589 & -0.0490 & 0.1589 \\ 0 & -0.1595 & -0.0963 \\ 0 & 0.0173 & -0.1319 \end{pmatrix}$$

Step 2. The economy size QR factorization of CU : $[Q, R] = \text{qr}(CU, 0)$ gives

$$R = \begin{pmatrix} 0.1100 & -0.0289 & 0.4245 \\ 0 & 0.1159 & 0.3260 \end{pmatrix}$$

Step 3. Partitioning of R and S gives

$$S_{11} = (-0.3589), \quad R_{11} = (0.1100)$$

which enables us to compute $Y_{11} = (0.1178)$.

Step 4. Dimension of $Y_{11} = 1 <$ dimension of $S = 3$. So we do:

Step 4.1.

$$s = \begin{pmatrix} -0.0490 \\ 0.1589 \end{pmatrix}^T, \quad s_1 = \begin{pmatrix} -0.1595 & -0.0963 \\ 0.0173 & -0.1319 \end{pmatrix}$$

$$r = \begin{pmatrix} -0.0289 \\ 0.4245 \end{pmatrix}^T, \quad r_1 = (0.1159 \ 0.3260)$$

$$\text{Step 4.2. } y = \begin{pmatrix} -0.0291 \\ 0.4078 \end{pmatrix}^T$$

Step 4.3. Solve for upper triangular y_1 with positive diagonal:

$$y_1 = \begin{pmatrix} 0.1167 & 0.3242 \\ 0 & 0.1392 \end{pmatrix}$$

Step 4.4. Form Y_{11}

$$Y_{11} = \begin{pmatrix} 0.1178 & -0.0291 & 0.4078 \\ 0 & 0.1167 & 0.3242 \\ 0 & 0 & 0.1392 \end{pmatrix}$$

and the loop in **Step 4** ends.

Step 5. Find the QR factorization of $Y_{11}U^T$: $[Q_1, \tilde{Y}] = qr(Y_{11}U^T)$ to obtain \tilde{Y} :

$$\tilde{Y} = \begin{pmatrix} -0.2034 & -0.0618 & -0.4807 \\ 0 & -0.1417 & -0.1355 \\ 0 & 0 & -0.0664 \end{pmatrix}$$

Step 6. Compute the solution:

$$Y = \begin{pmatrix} 0.2034 & 0.0618 & 0.4807 \\ 0 & 0.1417 & 0.1355 \\ 0 & 0 & 0.0664 \end{pmatrix}$$

MATCONTROL NOTE: Algorithm 8.6.2 has been implemented in MATCONTROL function **lyapchld**.

8.7 Comparisons of Different Methods and Conclusions

The analytical methods such as the ones based on evaluating the integral

$X = \int_0^\alpha e^{A^T t} C e^{At} dt$ for the Lyapunov equation, evaluating the infinite sum $\sum (A^k)^T C A^k$ for the discrete Lyapunov equation, and the finite series methods for the Sylvester and Lyapunov equations are not practical for numerical computations.

The methods, based on the reduction to Jordan and companion forms, will give inaccurate solutions when the transforming matrices are ill-conditioned. **The methods based on the reduction to Jordan and companion forms, therefore, in general should be avoided for numerical computations.**

From numerical viewpoints, the methods of choice are:

- The **Schur method (Section 8.5.2)** for the Lyapunov equation: $XA + A^T X = C$.
- The **Hessenberg-Schur method (Algorithm 8.5.1)** for the Sylvester equation: $XA + BX = C$.

- The **Schur method** (**Section 8.5.4**) for the discrete Lyapunov equation: $A^T X A - X = C$
- The **modified Schur methods** (**Algorithms 8.6.1 and 8.6.2**) for the Cholesky factors of the Lyapunov equation: $X A + A^T X = -C^T C$ and the discrete-Lyapunov equation: $A^T X A + C^T C = X$.

8.8 Some Selected Software

8.8.1 MATLAB CONTROL SYSTEM TOOLBOX

Matrix equation solvers.

`lyap` - Solve continuous Lyapunov equations

`dlyap` - Solve discrete Lyapunov equations

8.8.2 MATCONTROL

<code>CONDSYLVC</code> -	Finding the condition number of the Sylvester equation problem
<code>LYAPCHLC</code> -	Finding the Cholesky factor of the positive definite solution of the continuous-time Lyapunov equation
<code>LYAPCHLD</code> -	Find the Cholesky factor of the positive definite solution of the discrete-time Lyapunov equation
<code>LYAPCSD</code> -	Solving discrete-time Lyapunov equation using complex-Schur decomposition of A
<code>LYAPFNS</code> -	Solving continuous-time Lyapunov equation via finite series method
<code>LYAPHESS</code> -	Solving continuous-time Lyapunov equation via Hessenberg decomposition
<code>LYAPRSC</code> -	Solving the continuous-time Lyapunov equation via real-Schur decomposition
<code>LYAPRSD</code> -	Solving discrete-time Lyapunov equation via real-Schur decompostion
<code>SEPEST</code> -	Estimating the <code>sep</code> function with triangular matrices
<code>SEPKR</code> -	Computing the <code>sep</code> function using Kronecker product
<code>SYLVHCSC</code> -	Solving the Sylvester equation using Hessenberg and complex Schur decompositions
<code>SYLVHCSD</code> -	Solving the discrete-time Sylvester equation using Hessenberg and complex-Schur decompositions
<code>SYLVHESS</code> -	Solving the Sylvester equation via Hessenberg decomposition
<code>SYLVHRSC</code> -	Solving the Sylvester equation using Hessenberg and real Schur decompositions
<code>SYLVHUTC</code> -	Solving an upper triangular Sylvester equation

8.8.3 CSP-ANM

Solutions of the Lyapunov and Sylvester matrix equations

- The Schur method for the Lyapunov equations is implemented as `LyapunovSolve` [a, b] `SolveMethod → SchurDecomposition`] (continuous-time case) and `DiscreteLyapunovSolve` [a, b , `SolveMethod → SchurDecomposition`] (discrete-time case).
- The Hessenberg-Schur method for the Sylvester equations is implemented as `LyapunovSolve` [a, b, c , `SolveMethod → HessenbergSchur`] (continuous-time case) and `Discrete LyapunovSolve` [a, b, c , `SolveMethod → HessenbergSchur`] (discrete-time case).
- The Cholesky factors of the controllability and observability Gramians of a stable system are computed using `CholeskyFactorControllabilityGramian` [`system`] and `CholeskyFactorObservabilityGramian` [`system`].

8.8.4 SLICOT

Lyapunov Equations

<code>SB03MD</code>	Solution of Lyapunov equations and separation estimation
<code>SB03OD</code>	Solution of stable Lyapunov equations (Cholesky factor)
<code>SB03PD</code>	Solution of discrete Lyapunov equations and separation estimation
<code>SB03QD</code>	Condition and forward error for continuous Lyapunov equations
<code>SB03RD</code>	Solution of continuous Lyapunov equations and separation estimation
<code>SB03SD</code>	Condition and forward error for discrete Lyapunov equations
<code>SB03TD</code>	Solution of continuous Lyapunov equations, condition and forward error estimation
<code>SB03UD</code>	Solution of discrete Lyapunov equations, condition and forward error estimation

Sylvester Equations

<code>SB04MD</code>	Solution of continuous Sylvester equations (Hessenberg-Schur method)
<code>SB04ND</code>	Solution of continuous Sylvester equations (one matrix in Schur form)
<code>SB04OD</code>	Solution of generalized Sylvester equations with separation estimation
<code>SB04PD</code>	Solution of continuous or discrete Sylvester equations (Schur method)
<code>SB04QD</code>	Solution of discrete Sylvester equations (Hessenberg-Schur method)
<code>SB04RD</code>	Solution of discrete Sylvester equations (one matrix in Schur form)

Generalized Lyapunov Equations

<code>SG03AD</code>	Solution of generalized Lyapunov equations and separation estimation
<code>SG03BD</code>	Solution of stable generalized Lyapunov equations (Cholesky factor)

8.8.5 MATRIX_X

Purpose: Solve a discrete Lyapunov equation.

Syntax: `P=DLYAP (A, Q)`

Purpose: Solve a continuous Lyapunov equation.

Syntax: $P = \text{LYAP}(A, Q)$

8.8.6 LAPACK

The Schur method for the Sylvester equation $XA + BX = C$, can be implemented in LAPACK by using the following routines in sequence: GEES to compute the Schur decomposition, GEMM to compute the transformed right hand side, TRSYL to solve the (quasi-)triangular Sylvester equation and GEMM to recover the solution X .

8.9 Summary and Review

I. Applications

The applications of the Lyapunov equations include

- Stability and robust stability analyses (**Chapter 7**).
- Computations of the controllability and observability Grammians for stable systems (needed for internal balancing and model reduction) (**Chapter 14**).
- Computations of the H_2 and H_∞ norms (**Chapter 7 and Chapter 10**).
- Implementation of Newton's methods for Riccati equations (**Chapter 13**).

The applications of the Sylvester equations include

- Design of Luenberger observer (**Chapter 12**)
- Block-diagonalization of a matrix by similarity transformation.

II. Existence and Uniqueness Results

- (1) The Sylvester equation $XA + BX = C$ has a unique solution if and only if A and $-B$ do not have an eigenvalue in common (**Theorem 8.2.1**).
- (2) The Lyapunov equation $XA + A^T X = C$ has a unique solution if and only if A and $-A$ do not have an eigenvalue in common (**Corollary 8.2.1**).
- (3) The discrete Lyapunov equation $A^T X A - X = C$ has a unique solution if and only if the product of any two eigenvalues of A is not equal to 1 or A does not have an eigenvalue of modulus 1 (**Theorem 8.2.2**).

III. Sensitivity Results

$$(1) \ sep(B, -A) \text{ defined by } sep(B, -A) = \min_{X \neq 0} \frac{\|XA + BX\|_F}{\|X\|_F} = \sigma_{\min}(P),$$

where $P = I_n \otimes B + A^T \otimes I_m$, and m and n are, respectively, the orders of B and A , plays an important role in the sensitivity analysis of the Sylvester equation $XA + BX = C$ (**Theorem 8.3.1**).

- (2) $\text{sep}(A^T, -A)$ has an important role in the sensitivity analysis of the Lyapunov equation: $XA + A^T X = C$ (**Corollary 8.3.2**).
- (3) $\text{sep}_d(A^T, A) = \sigma_{\min}(A^T \otimes A^T - I_{n^2})$ has an important role in the sensitivity analysis of the discrete Lyapunov equation $A^T X A - X = C$ (**Theorem 8.3.4**).
- (4) If A is stable, then the sensitivity of the Lyapunov equation can be determined by solving the Lyapunov equation $HA + A^T H = -I$. $\|H\|_2$ is an indicator of the sensitivity of the stable Lyapunov equation $XA + A^T X = -C$ (**Theorem 8.3.3**).
- (5) If A and B are ill-conditioned, then the Sylvester equation $XA + BX = C$ is ill-conditioned (**Theorem 8.3.6**). Thus, if A is ill-conditioned, then the Lyapunov equation is also ill-conditioned. **But the converse is not true in general.**

IV. Sep-Estimation

The LINPACK style algorithm (**Algorithm 8.3.1**) gives an estimate of $\text{sep}(A, B)^T$ without computing the Kronecker product sum P , which is computationally quite sensitive.

V. Methods for solving the Lyapunov and Sylvester Equations

The analytical methods such as the finite-series method or the method based on evaluation of the integral involving the exponential matrix are not practical for numerical computations (**Section 8.4**).

The methods based on reduction to the Jordan canonical form and the companion form of a matrix should be avoided (**Section 8.5**).

The Schur methods for the Lyapunov equations (**Sections 8.5.2 and 8.5.4**) and the Hessenberg-Schur method (**Algorithms 8.5.1 and Section 8.5.7**) for the Sylvester equations are by far the best for numerical computations.

If only the Cholesky factors of stable Lyapunov equations are needed, the modified Schur methods (**Algorithms 8.6.1 and 8.6.2**) should be used. These algorithms compute the Cholesky factors of the solutions without explicitly computing the solutions themselves. The algorithms are numerically stable.

8.10 Chapter Notes and Further Reading

The results on the existence and uniqueness of the Lyapunov and Sylvester equations are **classical**. For proofs of these results, see Horn and Johnson (1991), Lancaster and Rodman (1995). See also Barnett and Cameron (1985), and Barnett and Storey (1970). The sensitivity issues of these equations and the perturbation results given in Section 8.3 can be found in Golub, Nash, and Van Loan (1979) and in Higham (1996).

The sensitivity result of the stable Lyapunov equation is due to Hewer and Kenney (1988). The sensitivity result of the stable discrete Lyapunov equation is due to Gahinet, Laub, Kenney, and Hewer (1990). The perturbation result of the discrete Lyapunov equation appears in

Petkov, Christov, and Konstantinov (1991). The results relating the ill-conditioning of the Sylvester equation and eigenvalues can be found in Ghavimi and Laub (1995). The LINPACK-style sep-estimation algorithm is due to Byers (1984). See Kågstrom and Poromaa (1996) for LAPACK-style algorithms. For perturbation results on generalized Sylvester equation, see Kågström (1994), Edelman et al. (1997, 1999). For description of LAPACK, see Anderson et al. (1999).

The Schur method for the Lyapunov equation is due to Bartels and Stewart (1972). The Schur method for the discrete Lyapunov equation is due to Barraud (1977). Independently of Barraud, a similar algorithm was developed by Kitagawa (1977). The Hessenberg-Schur algorithms for the Sylvester and discrete Sylvester equations are due to Golub, Nash, and Van Loan (1979). A good account of the algorithmic descriptions and implementational details of the methods for solving the discrete-Lyapunov equations appears in the recent book of Sima (1996).

The Cholesky-factor algorithms for the stable Lyapunov equations are due to Hammarling (1982). The Hessenberg algorithm for the Sylvester equation is due to Datta and Datta (1976) and Kreisselmeier (1972). For numerical solutions of the generalized Sylvester equation $AXB^T + CXD^T = E$, see Gardiner, Laub, Amato, and Moler (1992). For applications of generalized Sylvester equations of the above type including the computation of stable eigendecompositions of matrix pencils see Demmel and Kågström (1987, 1993a, 1993b), Kågström and Westin (1989), etc. See Kagström and Poromaa (1989, 1992) for block algorithms for triangular Sylvester equation (with condition estimator).

EXERCISES

1. Prove that the equation $A^*XB + B^*XA = -C$ has a unique solution X if and only if $\lambda_i + \bar{\lambda}_j \neq 0$ for all i and j , where λ_i is an eigenvalue of the generalized eigenvalue problem: $Ax = \lambda Bx$. (Here $A^* = (\bar{A})^T$ and $B^* = (\bar{B})^T$).
2. Let A be a normal matrix with $\lambda_1, \dots, \lambda_n$ as the eigenvalues. Then show that $\frac{\max_i |\lambda_i|}{\min_{ij} |\bar{\lambda}_i + \lambda_j|}$ can be regarded as the condition number of the Lyapunov equation $XA + A^*X = -C$, where $A^* = (\bar{A})^T$. Using the result, construct an example of an ill-conditioned Lyapunov equation.
3. If $A = (a_{ij})$ and $B = (b_{ij})$ are upper triangular matrices of order $m \times m$ and $n \times n$ respectively, then show that $X = (x_{ij})$ satisfying the Sylvester equation $AX + XB = C$ can be found from $x_{ij} = \frac{c_{ij} - \sum_{k=i+1}^m a_{ik}x_{kj} - \sum_{k=1}^{n-1} x_{ik}b_{kj}}{a_{ii} + b_{jj}}$.
4. Prove Theorems 8.3.1 and 8.3.4.
5. Using the perturbation results in Section 8.3, construct an example to show that the Sylvester equation problem $XA + BX = C$ can be very well-conditioned even when the eigenvector matrices for A and B are ill-conditioned.
6. Prove or disprove that if A and $-B$ have close eigenvalues, then the Sylvester equation $XA + BX = C$ is ill-conditioned.
7. Construct an 2×2 example to show that the bound (8.3.7) can be much smaller than the bound (8.3.3).
8. Derive the expression ϕ for the **condition number of the Lyapunov equation** given in Section 8.3.4.
9. Using the definition of the sep function, prove that if X is a unique solution of the Sylvester equation $XA + BX = C$, then $\|X\|_F \leq \frac{\|C\|_F}{sep(B, -A)}$.
10. Let $U^T AU = T = \begin{pmatrix} T_{11} & T_{12} & \cdots & T_{1p} \\ 0 & T_{22} & \cdots & T_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & T_{pp} \end{pmatrix}$ be the real Schur form of A , and assume that T_{11}, \dots, T_{pp} have disjoint spectra.
 - (a) Develop an algorithm to transform T to the block diagonal form:

$$Y^{-1}TY = diag(T_{11}, \dots, T_{pp}),$$

based on the solution of a Sylvester equation.

- (b) Show that if the spectra of the diagonal blocks of T are not distinctly separated, then there will be a substantial loss of accuracy (consult Bavey and Stewart (1979)).
 - (c) Construct an example to support the statement in (b).
 - (d) Develop an algorithm to compute e^{At} based on the block diagonalization of A .
11. Construct a simple example to show that the Cholesky factor L of the solution matrix $X = L^T L$ of the Lyapunov equation
- $$XA + A^T X = BB^T,$$
- where A is a stable matrix, is less sensitive (with respect to perturbations in A) than X .
- 12. Construct your own example to show that the Lyapunov equation $XA + A^T X = -C$ is always ill-conditioned if A is ill-conditioned with respect to inversion, but the converse is not true.
 - 13. Repeat the last exercise with the Sylvester equation $XA + BX = C$; that is, construct an example to show that the Sylvester equation $XA + BX = C$ will be ill-conditioned if both A and B are ill-conditioned, but the converse is not true.
 - 14. (a) Let A be a stable matrix. Show that the Lyapunov equation $XA + A^T X = -C$ can still be ill-conditioned if A has one or more eigenvalues close to the imaginary axes.
(b) Construct an example to illustrate the result in (a).
 - 15. Give an example to show that the backward error for the Sylvester equation $XA + BX = C$, where only A and B are perturbed, is large if an approximate solution Y of the equation is ill-conditioned.
 - 16. Give an example to illustrate that the backward error of an approximate solution to the Sylvester equation $XA + BX = C$ can be large, even though the relative residual is quite small.
 - 17. Prove that $\text{sep}(A, -B) > 0$ if and only if A and $-B$ do not have common eigenvalues.
 - 18. Let $K = I \otimes A^T + A^T \otimes I$ and $L = I \otimes S^T + S^T \otimes I$ be the Kronecker matrices, respectively, associated with the equations

$$XA + A^T X = -C$$

and

$$\hat{X}S + S^T \hat{X} = -\hat{C}$$

where $S = U^T AU$ is the real Schur form of A , and

$$\hat{C} = U^T CU.$$

- (a) Prove that $\|K^{-1}\|_2 = \|L^{-1}\|_2$

- (b) Using the result in (a), find a bound for the error, when A is only perturbed, in terms of the norm of the matrix A and the norm of L^{-1} .
- (c) Based on (a), and (b), develop an algorithm for estimating $\text{sep}(A^T, -A)$, analogous to the Byers' algorithm (Byers (1984)) for estimating $\text{sep}(A, B)$.

19. (**Relationship of the Distance to Instability and $\text{sep}(A)$**) (Van Loan (1985))

Define $\text{sep}(A) = \min\{\|AX + XA^*\|_F \mid X \in \mathbb{C}^{n \times n}, \|X\|_F = 1\}$

Then prove that

- (a) $\text{sep}(A) = 0$ if and only if A has an eigenvalue on the imaginary axis.
- (b) $\frac{1}{2}\text{sep}(A) \leq \beta(A) \leq \sigma_{\min}(A)$, where $\beta(A)$ is the distance to instability (see Chapter 7).
- (**Hint:** $\text{sep}(A) = \sigma_{\min}(I \otimes A + A \otimes I)$, and $\|B \otimes C\|_2 \leq \|B\|_2 \|C\|_2$).

20. Construct an example of an ill-conditioned discrete Lyapunov equation based on Theorem 8.3.4.

21. Prove that if $p(x)$ is a real polynomial of degree n having no pair of roots conjugate with respect to the unit circle, and T is the lower companion matrix of $p(x)$, then the unique solution X of the discrete-time equation

$$X - T^T X T = \text{diag}(1, 0, \dots, 0)$$

can be written explicitly as:

$$X = (I - \phi(S)^T \phi(S))^{-1},$$

where

$$S = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix},$$

$$\text{and } \phi(x) = \frac{p(x)}{x^n p(\frac{1}{x})}.$$

Discuss the numerical difficulties of using this method for solving the discrete Lyapunov equation.

Work out an example to demonstrate the difficulties.

22. Develop an algorithm, analogous to Algorithm 8.6.1, to find the Cholesky factor of the symmetric positive definite solution of the Lyapunov equation

$$AX + XA^T = -BB^T,$$

where B is $n \times m$ and has full rank.

23. Compare the flop-count of the real-Schur method and the complex-Schur method for solving the Lyapunov equation: $XA + A^TX = -C$.
24. Work out the flop-count of the Schur method for the discrete Lyapunov equation described in Section 8.5.4.
25. Construct an example to show that a small relative residual in the Sylvester equation does not guarantee a small backward error.
26. Repeat the last example with the Lyapunov equation, that is, construct an example to show that a small relative residual in the Lyapunov equation does not guarantee a small backward error.
27. Develop a method for solving the Sylvester equation

$$XA + BX = C,$$

based on the reduction of A to a companion form. Construct an example to demonstrate the numerical difficulties with this method.

28. Develop a method to solve the Lyapunov equation

$$A^TXA - X = -C$$

based on the reduction of A to a companion form. Construct an example to show that the algorithm may not be numerically effective.

29. Establish the round-off error bound (8.5.18):

$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} \leq \frac{cm\mu}{sep_d(A^T, A)}$$

for the Schur method to solve the discrete-Lyapunov equation (8.5.12).

30. Develop a Hessenberg-Schur algorithm to solve the discrete Sylvester equation

$$BXA + C = X$$

31. Develop an algorithm to solve the Sylvester equation $XA + BX = C$, based on the reductions of both A and B to real Schur forms.

Give a flop-count of this algorithm and compare this with that of Algorithm 8.5.1.

RESEARCH PROBLEMS

1. Devise an algorithm for solving the equation

$$A^T XB + B^T X A = -C$$

based on the **generalized real Schur-decomposition** of the pair (A, B) , described in Chapter 4.

2. Devise an algorithm for solving the equation

$$AXB + LXC = D$$

using the **generalized real Schur decomposition** of the pairs (A, L) and (C^T, B^T) .

3. Investigate if and how the norm of the solution of the discrete-stable Lyapunov equation

$$A^T X A - X = -I$$

provides information on the sensitivity of the discrete Lyapunov equation:

$$A^T X A - X = C.$$

4. (**Higham (1996)**). Derive conditions for the Sylvester equation $XA + BX = C$ to have a well-conditioned solution.

References

1. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK User's Guide*, 3rd Edition, SIAM, Philadelphia, 1999.
2. S. Barnett and R. G. Cameron, *Introduction to Mathematical Control Theory*, 2nd Edition, Clarendon Press, Oxford, 1985.
3. S. Barnett and C. Storey, *Matrix Methods in Stability Theory*, Nelson, London, 1970.
4. A. Y. Barraud, A numerical algorithm to solve $A^T X A - X = Q$, *IEEE Trans. Autom. Control*, AC-22, pp. 883-885, 1977.
5. R. H. Bartels and G. W. Stewart, Algorithm 432: solution of the matrix equation $AX + XB = C$, *Comm. ACM*, vol. 15, pp. 820-826, 1972.
6. C. A. Bavey and G. W. Stewart, An algorithm for computing reducing subspaces by block diagonalization, *SIAM J. Numer. Anal.*, vol. 16, pp. 359-367, 1979.
7. R. Byers, A LINPACK-style condition estimator for the equation $AX - XB^T = C$, *IEEE Trans. Autom. Control*, AC-29, pp. 926-928, 1984.
8. B. N. Datta and K. Datta, An algorithm to compute the powers of a Hessenberg matrix and it's applications, *Lin. Alg. Appl.* vol. 14, pp. 273-284, 1976.
9. B. N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Co., Pacific Grove, CA, 1995.
10. K. Datta, Y.P. Hong and R.B. Lee, Applications of linear transformation to matrix equations, *Lin. Alg. Appl.*, vol. 267, pp. 221-240, 1997.
11. E. DeSouza and S.P. Bhattacharyya, Controllability, observability and the solution of $AX - XB = C$, *Lin. Alg. Appl.*, vol. 39, pp. 167-188, 1981.
12. J. Demmel and Bo Kågström, Computing stable eigendecompositions of matrix pencils, *Lin. Alg. Appl.*, vol. 88/89, pp. 139-186, 1987.
13. J. Demmel and B. Kågström, The generalized Schur decomposition of an arbitrary pencil $A - \lambda B$: Robust software with error bounds and applications, Part I: Theory and algorithms, *ACM Trans. Math. Software*, vol. 19, no. 2, pp. 160-174, 1993a.
14. J. Demmel and B. Kågström, The generalized Schur decomposition of an arbitrary pencil $A - \lambda B$: Robust software with error bounds and algorithms, Part II: Theory and algorithms, *ACM Trans. Math. Software*, vol. 19, no. 2, pp. 175-201, 1993b.

15. A. Edelman, E. Elmroth and B. Kågström, A geometric approach to perturbation theory of matrices and matrix pencils, Part I: Versal deformations, *SIAM J. Matrix Anal. Appl.*, vol. 18, no. 3, pp. 653-692, 1997.
16. A. Edelman, E. Elmroth, and B. Kågström, A geometric approach to perturbation theory of matrices and matrix pencils, Part II: A stratification-enhanced staircase algorithm, *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 3, pp. 667-699, 1999.
17. P.M. Gahinet, A.J. Laub, C.S. Kenney, and G. Hewer, Sensitivity of the stable discrete-time Lyapunov equation, *IEEE Trans. Autom. Control*, vol. 35, pp. 1209-1217, 1990.
18. J.D. Gardiner, A.J. Laub, J.J. Amato, and C.B. Moler, Solution of the Sylvester matrix equation $AXB^T + CXD^T = E$, *ACM Trans Math. Software*, vol. 8, pp. 223-231, 1992.
19. J.D. Gardiner, M.R. Wette, A.J. Laub, J.J. Amato, and C.B. Moler, Algorithm 705: A FORTRAN-77 Software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$, *ACM Trans. Math. Software*, vol. 18, pp. 232-238, 1992.
20. A.R. Ghavimi and A.J. Laub, An implicit deflation method for ill-conditioned Sylvester and Lyapunov equations, *Num. Lin. Alg. Appl.*, vol. 2, pp. 29-49, 1995.
21. G.H. Golub, S. Nash and C.F. Van Loan, A Hessenberg-Schur method for the problem $AX + XB = C$, *IEEE Trans. Autom. Control*, AC-24, pp. 909-913, 1979.
22. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Edition, Johns Hopkins University, Baltimore, MD, 1996.
23. G.H. Golub and J.H. Wilkinson, Ill-conditioned eigensystems and the computation of the Jordan canonical form, *SIAM Rev.*, vol. 18, pp. 578-619, 1976.
24. S.J. Hammarling, Numerical solution of the stable nonnegative definite Lyapunov equation, *IMA J. Numer. Anal.*, vol. 2, pp. 303-323, 1982.
25. J.Z. Hearon, Nonsingular solutions of $TA - BT = C$, *Lin. Alg. Appl.*, vol. 16, pp. 57-63, 1977.
26. G. Hewer and C. Kenney, The sensitivity of the stable Lyapunov equation, *SIAM J. Contr. Optimiz.*, vol. 26, pp. 321-344, 1988.
27. N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM Philadelphia, 1996.
28. R.A. Horn and C.R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.
29. A. Jameson, Solution of the equation $AX + XB = C$ by the inversion of an $M \times M$ or $N \times N$ matrix, *SIAM J. Appl. Math.* vol. 66, pp. 1020-1023, 1968.

30. B. Kågström, A perturbation analysis of the generalized Sylvester equation ($AR-LB$, $DR-LE$) = (C, F), *SIAM J. Matrix Anal. Appl.*, vol. 15, no. 4, pp. 1045-1060, 1994.
31. B. Kågström and P. Poromaa, Distributed block algorithms for the triangular Sylvester equation with condition estimator, *Hypercube and Distributed Computers*, Elsevier Science Publishers, B.V. North Holland, pp. 233-248, 1989, (F. Andre and J.P. Verjus, Editors).
32. B. Kågström and P. Poromaa, Distributed and shared memory block algorithms for the triangular Sylvester equation with sep^{-1} estimators, *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 1, pp 90-101, 1992.
33. B. Kågström and P. Poromaa, LAPACK-style algorithms and software for solving the generalized Sylvester equation and estimating the separation between regular matrix pairs, *ACM Trans. Math. Software*, vol. 22, no. 1, pp. 78-103, 1996.
34. B. Kågström and L. Westin, Generalized Schur methods with condition estimators for solving the generalized Sylvester equation, *IEEE Trans. Automat. Control*, AC-34, no. 7, pp. 745-751, 1989.
35. T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.
36. G. Kitagawa, An algorithm for solving the matrix equation $X = FXF^T + S$, *Int. J. Control*, vol. 25, no. 5, pp. 745-753, 1977.
37. M.M. Konstantinov, N.D. Christov and P. Petkov, Perturbation analysis of linear control problems, *Prepr. IFAC 10th Congress*, Munich, 27-31 July, vol. 9, pp. 16-21, Pergamon Press, Oxford, 1987.
38. G. Kreisselmeier, A Solution of the bilinear matrix equation $AY + YB = -Q$, *SIAM J. Appl. Math.*, vol. 23, pp. 334-338, 1972.
39. P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, UK, 1995.
40. P. Petkov, N.D. Christov and M.M. Konstantinov, *Computational Methods for Linear Control Systems*, Prentice Hall, London, 1991.
41. V. Sima, *Algorithms for Linear-Quadratic Optimization*, Marcel Dekker, New York, 1996.
42. G. Starke and W. Niethammer, SOR for $AX - XB = C$, *Lin. Alg. Appl.*, vol. 154-156, pp. 355-375, 1991.
43. C.F. Van Loan, Using the Hessenberg decomposition in control theory, in *Algorithms and Theory in Filtering and Control, Mathematical Programming Study*, no. 8, North Holland, Amsterdam, pp. 102-111, 1982, (D.C. Sorensen and R.J. Wets, Editors).

44. C.F. Van Loan, How near is a stable matrix to an unstable matrix, *Contemporary Mathematics*, Amer. Math Soc., Providence, RI, vol. 47, pp. 465-477, 1985, (R. Brualdi et al., Editors).

PART III

CONTROL SYSTEMS DESIGN

Chapter 9. Realization and Subspace Identification

Chapter 10. Feedback Stabilization, Eigenvalue Assignment, and Optimal Control

Chapter 11. Numerical Methods and Conditioning of the Eigenvalue Assignment Problems

Chapter 12. State Estimation: Observer and the Kalman Filter

Chapter 13. Numerical Solutions and Conditioning of Algebraic Riccati Equations

Chapter 14. Internal Balancing and Model Reduction

Chapter 9

REALIZATION AND SUBSPACE IDENTIFICATION

Contents

9.1	Introduction	354
9.2	State-Space Realizations of a Transfer Function	355
9.2.1	Controllable and Observable Realizations	355
9.2.2	Minimal Realization	357
9.3	Computing Minimal Realizations From Markov Parameters	360
9.3.1	Some Basic Properties of the Hankel Matrix of Markov Parameters	361
9.3.2	An SVD Method for Minimal Realization	362
9.3.3	A Modified SVD Method for Minimal Realization	365
9.4	Subspace Identification Algorithms	370
9.4.1	A Subspace Deterministic Model Identification Algorithm	370
9.4.2	A Stochastic Subspace Model Identification Algorithm	375
9.4.3	Continuous-time System Identification	377
9.4.4	Frequency-Domain Identification	378
9.5	Some Selected Software	380
9.5.1	MATLAB CONTROL SYSTEM TOOLBOX	380
9.5.2	MATCONTROL	380
9.5.3	CSP-ANM	380
9.5.4	SLICOT	380
9.5.5	MATRIX _X	381
9.6	Summary and Review	381
9.7	Chapter Notes and Further Reading	382

Topics Covered

- State-space Realization of Transfer Function
- Minimal Realization
- Subspace Identifications (Time and Frequency Domain)

9.1 Introduction

In this Chapter, we consider the problems of state-space **realization** and **identification**.

The state-space realization problem is the problem to find the matrices A, B, C and D of the transfer function $G(s)$ in the continuous-time case or $G(z)$ in the discrete-time case, given a set of large number of Markov parameters.

In case of a discrete-time system, the Markov parameters can easily be computed from the input-output sequence of the systems (see **Exercise 5**). Finding Markov parameters in case of a continuous-time system is not that straightforward.

There may exist many realizations of the same transfer function matrix. Two such realizations, **controllable and observable realizations**, are obtained in **Section 9.2.1**.

A realization with the smallest possible dimension of A is called a **minimal realization**. A necessary and sufficient condition for a realization to be a minimal realization is established in **Theorem 9.2.1**, and it is shown in **Theorem 9.2.2** that two minimal realizations are related via a nonsingular transformation.

The existing algorithms for finding minimal realizations are all based on factoring the associated Hankel matrix (matrices) of Markov parameters. Some basic rank properties of these matrices, which are relevant to such factorizations, are established in **Section 9.3**.

Two **numerically viable algorithms** (**Algorithms 9.3.1 and 9.3.2**) based on the **singular value decomposition(s)** of these matrices are then described in Section 9.3. The algorithms are valid both for continuous-time and discrete-time state-space realizations provided the Markov parameters are known.

The identification problem is the problem of identifying system matrices A, B, C and D from a given set of input-output data, rather than Markov parameters.

Two time-domain subspace system identification algorithms (**Algorithms 9.4.1 and 9.4.2**) are presented in **Section 9.4**. These algorithms are based on the SVD decompositions of Hankel matrices constructed directly from the input-output sequences. The algorithms are presented for discrete-time systems identification; but can be used for identifying the continuous-time systems also, provided the first and higher derivatives of the inputs and outputs can be computed. In the last section (**Section 9.4.4**), we state a **frequency-domain subspace identification** algorithm (**Algorithm 9.4.3**). A frequency-domain state-space identification is concerned

with finding the system matrices, given a set of measured frequency responses. The algorithm is stated for identification of a continuous-time system; however, can be used for discrete-time identification also, with trivial modifications.

Reader's Guide

The readers familiar with material on state-space realization can skip Section 9.2 and Section 9.3.1.

9.2 State-Space Realizations of a Transfer Function

In this section, we show, given a transfer matrix, how to construct state-space realizations in controllable and observable forms of this transfer matrix.

We consider here only the continuous-time case. The results are also valid for the discrete-time case by replacing the variable s by the variable z .

Definition 9.2.1 *Let $G(s)$ be the transfer matrix of order $r \times m$ which is proper. Then the quadruple (A, B, C, D) such that*

$$G(s) = C(sI - A)^{-1}B + D \quad (9.2.1)$$

is called a state-space realization of $G(s)$.

It can be shown [Exercise 1] that given a proper rational function $G(s)$, there always exists a state-space realization of $G(s)$. However, **such a realization is not unique**; that is, there may exist many state-space realizations of the same transfer matrix.

Below we show the non-uniqueness of the state-space realization of a transfer matrix (for the single-input, single-output case (SISO)), by constructing two realizations of the same transfer matrix.

9.2.1 Controllable and Observable Realizations

The transfer matrix $G(s)$ can be written in the form

$$G(s) = D + \frac{P(s)}{d(s)}, \quad (9.2.2)$$

where $P(s)$ is a polynomial matrix in s of degree at most $h - 1$ given by

$$P(s) = P_0 + P_1s + \cdots + P_{h-1}s^{h-1}, \quad (9.2.3)$$

and $d(s) = s^h + d_{h-1}s^{h-1} + \cdots + d_1s + d_0$ is a monic polynomial of degree h (h is the least common multiple of the denominators of all the entries of $G(s)$).

Let 0_p and I_p denote, respectively, the zero and identity matrices of order p .

Define now

$$A = \begin{pmatrix} 0_m & I_m & & & \\ & 0_m & I_m & & \\ \vdots & & \ddots & \ddots & \\ 0_m & & \cdots & 0_m & I_m \\ -d_0 I_m & -d_1 I_m & -d_2 I_m & \cdots & -d_{h-1} I_m \end{pmatrix}, \quad (9.2.4)$$

$$B = \begin{pmatrix} 0_m \\ 0_m \\ \vdots \\ 0_m \\ I_m \end{pmatrix}, \quad C = (P_0, \dots, P_{h-1}), \quad (9.2.5)$$

$$D = \lim_{s \rightarrow \infty} G(s), \quad (9.2.6)$$

Then it is easily verified that

$$C(sI - A)^{-1}B + D = G(s) = D + \frac{P(s)}{d(s)}. \quad (9.2.7)$$

Since the matrix-pair (A, B) is controllable, the above realization of $G(s)$ is called a **controllable realization**. This realization has dimension mh .

We now construct a different realization of $G(s)$.

Expand $G(s)$ in Taylor series:

$$G(s) = D' + \frac{1}{s}H_1 + \frac{1}{s^2}H_2 + \dots \quad (9.2.8)$$

The matrices $\{D', H_k\}$ can be found as follows:

$$\begin{aligned} D' &= \lim_{s \rightarrow \infty} G(s) \\ H_1 &= \lim_{s \rightarrow \infty} s(G(s) - D') \\ H_2 &= \lim_{s \rightarrow \infty} s^2(G(s) - D' - \frac{1}{s}H_1) \\ &\vdots \\ \text{etc.} & \end{aligned} \quad (9.2.9)$$

Definition 9.2.2 The matrices $\{H_i\}$, defined above, are called the **Markov parameters of $G(s)$** .

Note: The Markov parameters $\{H_i\}$ can be expressed as:

$$H_i = CA^{i-1}B, \quad i = 1, 2, \dots \quad (9.2.10)$$

Define now the matrices A' , B' and C' as follows:

$$A' = \begin{pmatrix} 0_r & I_r & & & \\ & 0_r & I_r & & \\ \vdots & & \ddots & \ddots & \\ 0_r & \cdots & 0_r & I_r & \\ -d_0 I_r & -d_1 I_r & -d_2 I_r & \cdots & -d_{h-1} I_r \end{pmatrix}, \quad (9.2.11)$$

$$B' = \begin{pmatrix} H_1 \\ H_2 \\ H_3 \\ \vdots \\ H_h \end{pmatrix}, \quad C' = (I_r, 0_r, \dots, 0_r). \quad (9.2.12)$$

Then it can be shown that with A' , B' , C' , and D' as defined above, we have

$$C'(sI - A')^{-1}B' + D' = G(s). \quad (9.2.13)$$

That is, we have now another realization of $G(s)$. Since (C', A') is observable, this realization is called an **observable realization** of $G(s)$. This realization has dimension rh .

9.2.2 Minimal Realization

Since there may exist more than one realizations of the same transfer function $G(s)$, it is natural to look for a realization of minimal order.

Definition 9.2.3 A state-space realization (A, B, C, D) of $G(s)$ is said to be a minimal realization of $G(s)$ if the matrix A has the smallest possible dimension; that is, if (A', B', C', D') is any other realization of $G(s)$, then the order of A' is greater than or equal to the order of A . The dimension of a minimal realization is called the **McMillan degree**.

Theorem 9.2.1 A state-space realization (A, B, C, D) of $G(s)$ is minimal if and only if (A, B) is controllable and (A, C) is observable.

Proof: We first prove the necessity by **contradiction**.

If (A, B) is not controllable and/or (A, C) is not observable, then from Kalman decomposition (see **Chapter 6**), it follows that there exists a realization of smaller dimension that is both controllable and observable. This contradicts the minimality assumption.

Conversely, let (A, B, C, D) and (A', B', C', D') be two minimal realizations of $G(s)$. Assume that the order of A' is $n' < n$. Since the two realizations have the same transfer function, then they should have the same Markov parameters, that is,

$$CA^{i-1}B = C'(A')^{i-1}B'. \quad (9.2.14)$$

This implies that

$$O_M C_M = O'_M C'_M, \quad (9.2.15)$$

where O_M and C_M , respectively, denote the observability and controllability matrices of the realization (A, B, C, D) and, O'_M and C'_M , respectively, denote the observability and controllability matrices of the realization (A', B', C', D') .

But, $\text{rank}(O_M C_M) = n$, and $\text{rank}(O'_M C'_M) = n' < n$. This is a contradiction, since $\text{rank}(O_M C_M) = \text{rank}(O'_M C'_M)$, by (9.2.15). ■

The next question is how are two minimal realizations of the same transfer matrices related? We answer the question in the following Theorem.

Theorem 9.2.2 *If (A, B, C, D) and (A', B', C', D') are two minimal realizations of the same transfer function $G(s)$, then there exists a unique nonsingular matrix T such that*

$$A' = T^{-1} A T, \quad (9.2.16)$$

$$B' = T^{-1} B, \quad C' = C T, \quad D' = D. \quad (9.2.17)$$

Moreover, T is explicitly given by

$$T = (O_M^T O_M)^{-1} \cdot O_M^T O'_M \quad (9.2.18)$$

or

$$T = C_M (C'_M)^T [C'_M (C'_M)^T]^{-1}, \quad (9.2.19)$$

where C_M and O_M , are, respectively, the controllability and observability matrices of the realization (A, B, C, D) ; and C'_M , and O'_M are, respectively, the controllability and observability matrices, of the realization (A', B', C', D') .

Proof: We just sketch a proof here and leave the details to the readers.

Let T be the matrix relating the matrices O_M and O'_M ; that is T satisfies the matrix equation

$$O_M T = O'_M. \quad (9.2.20)$$

Since O_M has full rank, such a matrix T always exists. In fact, it is unique and is given by

$$T = (O_M^T O_M)^{-1} O_M^T O'_M. \quad (9.2.21)$$

From the first block row of the equation (9.2.20), we then have $CT = C'$.

Since both the realizations have the same transfer function, and hence the same Markov parameters, we obtain

$$O_M C_M = O'_M C'_M \quad (9.2.22)$$

which gives

$$C_M = (O_M^T O_M)^{-1} O_M^T O'_M C'_M = T C'_M. \quad (9.2.23)$$

That is, T is a solution of the equation

$$TC'_M = C_M \quad (9.2.24)$$

Since C'_M has full rank, we have

$$T = C_M(C'_M)^T[C'_M(C'_M)^T]^{-1}, \text{ establishing (9.2.19).}$$

Again, from the first block column of the the equation (9.2.23), we have

$$TB' = B. \quad (9.2.25)$$

All that remains to be shown is that (9.2.16) holds. To show this, first note that the Markov parameters $CA^{i-1}B$ and $C'(A')^{i-1}B'$, $i \geq 1$, are equal.

We can then write

$$O_M A C_M = O'_M A' C'_M \quad (9.2.26)$$

which leads to

$$O_M^T O_M A C_M = O_M^T O'_M A' C'_M \quad (9.2.27)$$

From (9.2.27) we have

$$AC_M = TA'C'_M, \text{ (where } T \text{ is defined by (9.2.18)).} \quad (9.2.28)$$

But again multiplying (9.2.19) by A to the left, we have

$$AC_M(C'_M)^T(C'_M(C'_M)^T)^{-1} = AT \quad (9.2.29)$$

From (9.2.28) and (9.2.29), we obtain

$$AT = TA'$$

That is, $A' = T^{-1}AT$.

■

Uniqueness. Suppose that there exists another similarity transformation given by T_1 relating both the systems. Then we must have

$$O_M(T - T_1) = 0.$$

But O_M has full rank; so, $T = T_1$.

■

Example 9.2.1 Let

$$G(s) = \frac{3s - 4}{s^2 - 3s + 2}$$

Here

$$\begin{aligned} P(s) &= -4 + 3s \\ d(s) &= s^2 - 3s + 2 \end{aligned} .$$

The Markov parameters are:

$$\begin{aligned} D' &= \lim_{s \rightarrow \infty} G(s) = 0. \\ H_1 &= \lim_{s \rightarrow \infty} s(G(s) - D') = 3 \\ H_2 &= \lim_{s \rightarrow \infty} s^2(G(s) - D' - \frac{1}{s}H_1) = 5. \end{aligned}$$

(I) Controllable realization

$$A = \begin{pmatrix} 0 & 1 \\ -2 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad C = (-4, 3), \quad D = 0.$$

Verify:

$$C(sI - A)^{-1}B = \frac{1}{s^2 - 3s + 2}(-4, 3) \begin{pmatrix} s-3 & 1 \\ -2 & s \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{3s-4}{s^2 - 3s + 2}.$$

Since (A, B) is controllable and (A, C) is observable the realization is a **minimal realization**.

(II) Observable Realization

$$\begin{aligned} A' &= \begin{pmatrix} 0 & 1 \\ -2 & 3 \end{pmatrix}, \quad B' = \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \\ C' &= (1, 0). \end{aligned}$$

$$\text{Verify: } C'(sI - A)^{-1}B' = \frac{3s-4}{s^2 - 3s + 2}$$

Since (A', B') is controllable, and (A', C') is observable, this is also a minimal realizations of $G(s)$.

(III) Relationship. The two realizations are related by the nonsingular transforming matrix T given by

$$T = (O_M^T O_M)^{-1} O_M^T O_M' = \begin{pmatrix} -2.5 & 1.5 \\ -3 & 2 \end{pmatrix}.$$

$$\text{Verify: } T^{-1}AT = \begin{pmatrix} 0 & 1 \\ -2 & 3 \end{pmatrix} = A', \quad T^{-1}B = \begin{pmatrix} 3 \\ 5 \end{pmatrix} = B', \quad CT = (1, 0) = C'.$$

9.3 Computing Minimal Realizations From Markov Parameters

In the last section, we showed how to obtain an observable realization from a set of Markov parameters:

$$H_k = CA^{k-1}B, k = 1, 2, \dots$$

Here we consider the problem of computing a minimal realization (MR) using Markov parameters. Specifically, the following problem is considered.

Given a set of large number Markov parameters $\{H_k\}$ of an unknown transfer function $G(s)$, find a minimal realization (A, B, C, D) whose transfer function $G(s) = C(sI - A)^{-1}B + D$.

Since the Markov parameters are much easier to obtain for a discrete-time system, unless otherwise stated, we assume that the given Markov parameters are of the discrete-time system:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k. \end{aligned} \quad (9.3.1)$$

9.3.1 Some Basic Properties of the Hankel Matrix of Markov Parameters

There exist many methods for finding a minimal realization (see DeJong (1978) for a survey). Most of these methods find a minimal realization from a decomposition or a factorization of the Hankel matrix of Markov parameters of the form:

$$M_k = \begin{pmatrix} H_1 & H_2 & \cdots & H_k \\ H_2 & H_3 & \cdots & H_{k+1} \\ \vdots & \vdots & & \vdots \\ H_k & H_{k+1} & \cdots & H_{2k-1} \end{pmatrix}. \quad (9.3.2)$$

For example, a recursive method due to Rissanen (1971) obtains a minimal realization by recursively updating the decomposition of a smaller Hankel matrix to that of a larger Hankel matrix.

The following basic results due to Kalman et al. (see, e.g., Kalman, Falb and Arbib (1969)), play an important role in the developments of Rissanen's and other methods.

Theorem 9.3.1 (i) $\text{rank}(M_k) \leq \text{rank}(M_{k+1})$ for all k .

(ii) If (A, B, C, D) is any realization of dimension n , then $\text{rank}(M_k) = \text{rank}(M_n)$ for all $k \geq n$.

(iii) Let (A, B, C, D) and (A', B', C', D') be two realizations of $G(s)$ of order n and n' , respectively. Then

$$\text{rank}(M_n) = \text{rank}(M'_n).$$

(iv) Let d be the McMillan degree, then $\max_k(\text{rank}(M_k)) = d$.

(v) Let (A, B, C, D) be any realization of dimension n , then

$$d = \text{rank}(M_n) = \text{rank}(O_M C_M),$$

where O_M and C_M are, respectively, the observability and controllability matrices of the realization (A, B, C, D) .

Proof: (i) The proof of (i) follows from the fact that M_k is a submatrix of M_{k+1} .

(ii) The proof of (ii) follows by observing that [Exercise 6] the Hankel matrix M_k can be decomposed as:

$$M_k = \begin{pmatrix} O_M \\ CA^n \\ \vdots \\ CA^{k-1} \end{pmatrix} \left(C_M | A^n B, A^{n+1} B, \dots, A^{k-1} B \right), \quad (9.3.3)$$

$$\text{for } k \geq n, \text{ and } M_n = O_M C_M. \quad (9.3.4)$$

Since the rows in (CA^n, \dots, CA^{k-1}) and the columns in $(A^n B, \dots, A^{k-1} B)$ are linear combination of the rows in O_M and the columns in C_M , respectively; we have

$$\text{rank}(M_k) = \text{rank}(M_n) = \text{rank}(O_M C_M).$$

(iii) Let (A', B', C', D') be another realization of $G(s)$ of order n' and let $r = \max(n, n')$. Then, since both these realization have the same Markov parameters, we must have

$$M'_r = M_r.$$

Thus by (ii), $\text{rank}(M_n) = \text{rank}(M_r) = \text{rank}(M'_r) = \text{rank}(M'_n)$.

(iv) The proof is by *contradiction*. Suppose that there exists a minimal realization (A, B, C, D) of order $d' < d$.

Then by the previous two results, we should have $\max(\text{rank}(M_k)) = d_1 < d'$, a contradiction.

(v) The proof follows from (iii) and (iv).

Finding the McMillan Degree

The above result gives us several alternative procedures to obtain the McMillan degree of the transfer function matrix.

A simple way to do so is to find any realization of $G(s)$ and then find the rank of the product $O_M C_M$, using the SVD.

Also, if the realization is stable and C_G and O_G are respectively, the controllability and observability Grammians obtained via solutions of respective Lyapunov equations (see Chapter 7), then it is well-known (Glover (1984)) that the McMillan degree is equal to the rank of $C_G O_G$.

9.3.2 An SVD Method for Minimal Realization

It was shown by DeJong (1978) that the **Rissanen's method is numerically unstable**.

Since the singular value decomposition provides a numerically reliable way to compute the rank of a matrix, a more numerically viable method for finding a MR should be based on the SVD of the associated Hankel matrix. We now describe below an SVD-based method for finding a MR (Ho and Kalman (1966), Zeiger and McEwen (1974), Kung (1978)). **For the sake of convenience, we will assume that $D = 0$ in this section.**

Given the set $\{H_1, H_2, \dots, H_{2N+1}\}$ of Markov parameters, consider the SVD of M_{N+1} :

$$M_{N+1} = USV^T = US^{\frac{1}{2}}S^{\frac{1}{2}}V^T = U'V',$$

where $S = \text{diag}(s_1, s_2, \dots, s_p, 0, \dots, 0)$ and $U' = US^{\frac{1}{2}}$ and $V' = S^{\frac{1}{2}}V^T$.

Comparing this decomposition with the decomposition of M_{N+1} in the form (9.3.2) in Section 9.3.1, it is easy to see that we can take C as the first block row and the first p columns of U' and similarly B can be taken as the first p rows and the first block column of V' .

The matrix A satisfies the relations:

$$U_1 A = U_2,$$

and

$$AV_1 = V_2,$$

where

U_1 = The first N block rows and the first p columns of U'

V_1 = The first p rows and the first N block columns of V' .

U_2 and V_2 are similarly defined. Since U_1 and V_1 have full ranks, we immediately have from the above two equations,

$$A = U_1^\dagger U_2$$

or

$$A = V_2 V_1^\dagger,$$

where U_1^\dagger and V_1^\dagger are the generalized inverses of U_1 and V_1 , respectively.

This discussion leads to the following SVD algorithm for finding a **minimal realization**:

Algorithm 9.3.1 (An SVD Algorithm for Minimal Realization)

Inputs: The set of Markov parameters: $\{H_1, H_2, \dots, H_{2N+1}\}$ (N should be at least equal to the McMillan degree).

Outputs: The matrices A, B , and C of a minimal realization.

Step 1. Find the SVD of the Hankel matrix

$$M_{N+1} = \begin{pmatrix} H_1 & H_2 & \cdots & H_{N+1} \\ H_2 & H_3 & \cdots & H_{N+2} \\ \vdots & & & \\ H_{N+1} & H_{N+2} & \cdots & H_{2N+1} \end{pmatrix} = USV^T,$$

where $S = \text{diag}(s_1, s_2, \dots, s_p, 0, \dots, 0)$, and $s_1 \geq s_2 \geq \dots \geq s_p > 0$

Step 2. Form $U' = US^{\frac{1}{2}}$ and $V' = S^{\frac{1}{2}}V^T$, where $S^{\frac{1}{2}} = \text{diag}(s_1^{\frac{1}{2}}, s_2^{\frac{1}{2}}, \dots, s_p^{\frac{1}{2}}, 0, \dots, 0)$.

Step 3. Define

$$U_1 = \text{The first } N \text{ block rows and the first } p \text{ columns of } U'$$

$$U_2 = \text{The last } N \text{ block rows and the first } p \text{ columns of } U'$$

$$U^{(1)} = \text{The first block row and the first } p \text{ columns of } U'$$

$$V^{(1)} = \text{The first } p \text{ rows and the first block column of } V'.$$

Step 4. Compute $A = U_1^\dagger U_2$, Set $B = V^{(1)}$, $C = U^{(1)}$.

■

Then the following Theorem due to Kung (1978) shows that the minimal realization obtained by Algorithm 9.3.1 enjoys certain desirable properties.

Theorem 9.3.2 Let E_i denote the error matrix; that is,

$$E_i = CA^{i-1}B - H_i, i \geq 1.$$

Assume that the given impulse response sequence $\{H_k\}$ is convergent. That is, $H_k \rightarrow 0$, when $k \rightarrow \infty$.

Then

(i) $\sum_{i=1}^{2N+1} \|E_i\|_F^2 \leq \epsilon \sqrt{n+m+r}$, where ϵ is a small positive number, and n, m and r are, respectively, the number of states, inputs and outputs.

(ii) The minimal realization obtained by Algorithm 9.3.1 is (a) **discrete-stable** and (b) **internally balanced**; that is, the controllability and observability Grammians for this realization are the same and are equal to a diagonal matrix.

Example 9.3.1 Let $N = 2$ and the given set of Markov parameters be:

$$\{H_1, H_2, H_3, H_4, H_5\} = \{3, 5, 9, 17, 33\}.$$

Step 1. $M_3 = \begin{pmatrix} 3 & 5 & 9 \\ 5 & 9 & 17 \\ 9 & 17 & 33 \end{pmatrix}$. Then $U = \begin{pmatrix} 0.2414 & -0.8099 & 0.5345 \\ 0.4479 & -0.3956 & -0.8018 \\ 0.8609 & 0.4330 & 0.2673 \end{pmatrix}$,

$$S = \text{diag}(44.3689 \ 0.6311 \ 0), \text{ and } V^T = \begin{pmatrix} 0.2414 & 0.4479 & 0.8609 \\ -0.8099 & -0.3956 & 0.4330 \\ 0.5345 & -0.8018 & 0.2673 \end{pmatrix}.$$

Step 2. $U' = \begin{pmatrix} 1.6081 & -0.64340 & 0 \\ 2.9835 & -0.31430 & 0 \\ 5.7343 & 0.34400 & 0 \end{pmatrix}$, $V' = \begin{pmatrix} 1.6081 & 2.9835 & 5.7343 \\ -0.6434 & -0.3143 & 0.3440 \\ 0 & 0 & 0 \end{pmatrix}$.

Step 3.

$$U_1 = \begin{pmatrix} 1.6081 & -0.6434 \\ 2.9835 & -0.3143 \end{pmatrix}$$

$$U_2 = \begin{pmatrix} 2.9835 & -0.3143 \\ 5.7343 & 0.3440 \end{pmatrix}.$$

$$U^{(1)} = (1.6081 \quad -0.6434)$$

$$V^{(1)} = \begin{pmatrix} 1.6081 \\ -0.6434 \end{pmatrix}.$$

Step 4.

$$\begin{aligned} A &= U_1^\dagger U_2 = \begin{pmatrix} 1.9458 & 0.2263 \\ 0.2263 & 1.0542 \end{pmatrix} \\ B &= V^{(1)} = \begin{pmatrix} 1.6081 \\ -0.6434 \end{pmatrix} \\ C &= U^{(1)} = (1.6081 \quad -0.6434). \end{aligned}$$

Verify:

$$\begin{aligned} E_1 &= CB - H_1 = -8.8818 \times 10^{-16} \\ E_2 &= CAB - H_2 = -8.8818 \times 10^{-16} \\ E_3 &= CA^2B - H_3 = -1.7764 \times 10^{-15} \\ E_4 &= CA^3B - H_4 = 0 \\ E_5 &= CA^4B - H_5 = 7.1054 \times 10^{-15} \\ \sum_{i=1}^5 |E_i|^2 &= 5.5220 \times 10^{-30}. \end{aligned}$$

It is also easy to check that the realization is both controllable and observable. So, it is minimal. The controllability and observability Grammians are the same and are given by: $C_G^N = O_G^N = \text{diag}(44.3689, \ 0.6311)$. **The realization is, therefore, internally balanced.**

The graph below shows a comparision between the graphs of the transfer functions $G_0(s) = \sum_{i=1}^5 \frac{1}{s^i} H_i$ and $G(s) = C(sI - A)^{-1}B$. The plot shows an excellent agreement between the graphs for large values for s .

MATCONTROL NOTES: Algorithm 9.3.1 has been implemented in MATCONTROL function **minresvd**.

9.3.3 A Modified SVD Method for Minimal Realization

We describe now a modification of the above algorithm (see Juang (1994)).

This modified algorithm uses **lower-order block Hankel matrices** in computing the system matrices A, B and C .

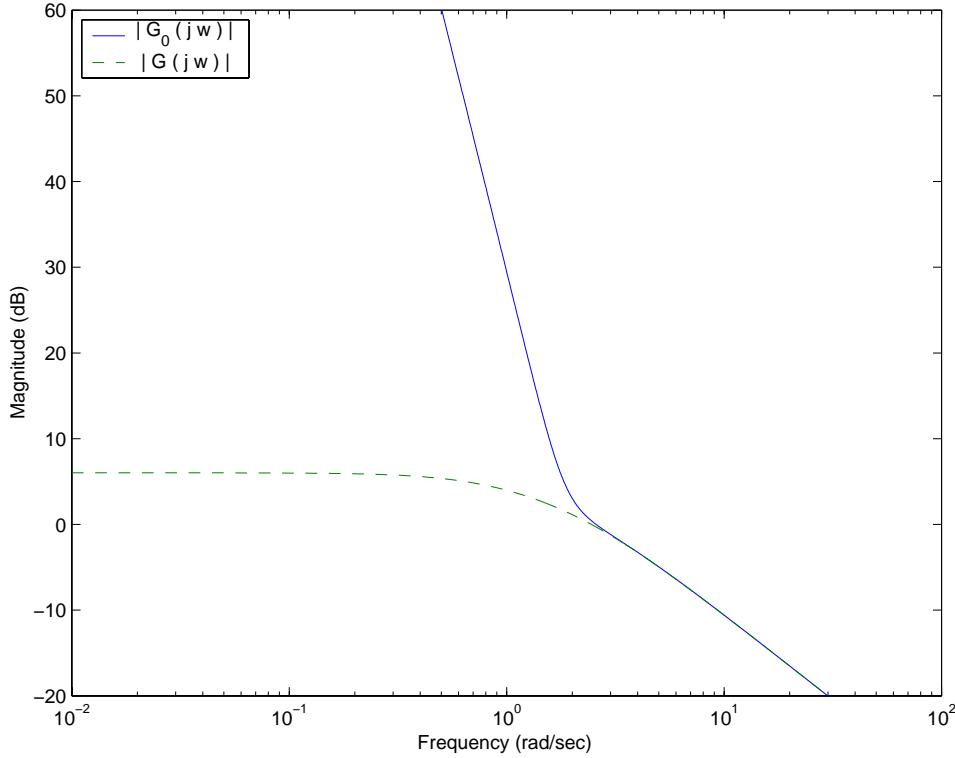


Figure 9.1: Comparison of Transfer Functions

Define the block Hankel matrices:

$$M_R = \begin{pmatrix} H_1 & H_2 & \cdots & H_R \\ H_2 & H_3 & \cdots & H_{R+1} \\ \vdots & \vdots & & \vdots \\ H_R & H_{R+1} & \cdots & H_{2R-1} \end{pmatrix}$$

and

$$M_{R1} = \begin{pmatrix} H_2 & H_3 & \cdots & H_{R+1} \\ H_3 & H_4 & \cdots & H_{R+2} \\ \vdots & \vdots & & \vdots \\ H_{R+1} & H_{R+2} & \cdots & H_{2R} \end{pmatrix},$$

where $R \geq n$ (n is the order of the system). Denote the controllability and observability matrices by:

$$C_M^R = (B, AB, \dots, A^{R-1}B)$$

and

$$O_M^R = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{R-1} \end{pmatrix}.$$

Then

$$M_{R1} = O_M^R A C_M^R,$$

and $M_R = O_M^R C_M^R$.

Consider now the SVD of M_R :

$$M_R = U \Sigma V^T = U \Sigma^{\frac{1}{2}} \Sigma^{\frac{1}{2}} V^T.$$

This means that O_M^R is related to U and C_M^R is related to V .

Define now Σ_n by: $\Sigma = \begin{pmatrix} \Sigma_n & 0 \\ 0 & 0 \end{pmatrix}$, and U_n and V_n as the matrices formed by the first n columns of U and V , respectively. Also, let the matrices E'_r and E'_m be defined as:

$$E'^T_r = (I_r, 0, \dots, 0), \quad E'^T_m = (I_m, 0, \dots, 0),$$

where I_s is an identity matrix of order s ; and m and r denote, respectively, the number of inputs and the number of outputs.

Then one can choose $O_M^R = U_n \Sigma_n^{\frac{1}{2}}$ and $C_M^R = \Sigma_n^{\frac{1}{2}} V_n^T$.

From the equation

$$M_R = O_M^R C_M^R = U_n \Sigma_n^{\frac{1}{2}} \Sigma_n^{\frac{1}{2}} V_n^T$$

it then follows that B and C can be chosen as:

$$B = \Sigma_n^{\frac{1}{2}} V_n^T E'_m \text{ and } C = E'^T_r U_n \Sigma_n^{\frac{1}{2}}.$$

Also, from the equation

$$M_{R1} = O_M^R A C_M^R = U_n \Sigma_n^{\frac{1}{2}} A \Sigma_n^{\frac{1}{2}} V_n^T$$

it follows that

$$A = \Sigma_n^{-\frac{1}{2}} U_n^T M_{R1} V_n \Sigma_n^{-\frac{1}{2}}$$

Thus we have the following modified algorithm using the SVD of lower-order Hankel matrices.

Algorithm 9.3.2 A Modified SVD Algorithm for Minimal Realization

Inputs: The Markov parameters $\{H_1, H_2, \dots, H_{2R}\}$, $R \geq n$ (the order of the system to be identified).

Outputs: A, B, C of a Minimal Realization.

Step 1. Form the Hankel matrices M_R and M_{R1} as defined above.

Step 2. Find the SVD of M_R :

$$M_R = U \begin{pmatrix} \Sigma_n & 0 \\ 0 & 0 \end{pmatrix} V^T,$$

where $\Sigma_n = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$; $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n > 0$.

Step 3. Compute

$$\begin{aligned} A &= \Sigma_n^{-\frac{1}{2}} U_n^T M_{R1} V_n \Sigma_n^{-\frac{1}{2}} \\ B &= \Sigma_n^{\frac{1}{2}} V_n^T E'_m \\ C &= E'^T_r U_n \Sigma_n^{\frac{1}{2}}, \end{aligned}$$

where U_n and V_n are the matrices of the first n columns of U and V , respectively; and E'_m and E'_r are as defined above.

Example 9.3.2 We consider the previous example again. Take $R = n = 2$. Let $m = 1, r = 1$.

Then

$$\begin{aligned} M_R &= \begin{pmatrix} H_1 & H_2 \\ H_2 & H_3 \end{pmatrix} = \begin{pmatrix} 3 & 5 \\ 5 & 9 \end{pmatrix}, \\ M_{R1} &= \begin{pmatrix} H_2 & H_3 \\ H_3 & H_4 \end{pmatrix} = \begin{pmatrix} 5 & 9 \\ 9 & 17 \end{pmatrix}. \end{aligned}$$

$$\Sigma_2 = \text{diag} \begin{pmatrix} 11.8310 & 0 \\ 0 & 0.1690 \end{pmatrix}$$

$$U_2 = \begin{pmatrix} 0.4927 & -0.8702 \\ 0.8702 & 0.4927 \end{pmatrix}$$

$$V_2 = \begin{pmatrix} 0.4927 & -0.8702 \\ 0.8702 & 0.4927 \end{pmatrix}.$$

$$A = \Sigma_2^{-\frac{1}{2}} U_2^T M_{R1} V_2 \Sigma_2^{-\frac{1}{2}} = \begin{pmatrix} 1.8430 & 0.3638 \\ 0.3638 & 1.1570 \end{pmatrix}, \quad B = \Sigma_2^{\frac{1}{2}} V_2^T E'_1 = \begin{pmatrix} 1.6947 \\ -0.3578 \end{pmatrix},$$

$$C = E'^T_1 U_2 \Sigma_2^{\frac{1}{2}} = (1.6947 \quad -0.3578).$$

Verification:

$$\begin{aligned} E_1 &= CB - H_1 = 0 \\ E_2 &= CAB - H_2 = -8.8818 \times 10^{-16} \\ E_3 &= CA^2B - H_3 = -3.5527 \times 10^{-15} \\ E_4 &= CA^3B - H_4 = 0 \\ E_5 &= CA^4B - H_5 = 0. \end{aligned}$$

Remarks:

- (i) Algorithm 9.3.2, when extended to reconstruct the Markov parameters of a reduced-order system obtained by eliminating “noisy modes”, is called **Eigensystem Realization Algorithm** (ERA); because, information from the eigensystem of the realized state matrix obtained in Algorithm 9.3.2 is actually used to obtain the reduced-order model. The details can be found in Juang (1994, pp. 133-144).
- (ii) The optimal choice of the number R requires engineering intuition. The choice has to be made based on measurement data to minimize the size of the Hankel matrix M_R . See Juang (1994).

The figure below shows a comparison between the graphs of the transfer function $G_0(s) = \sum_{i=1}^4 \frac{1}{s^i} H_i$ and $G(s) = C(sI - A)^{-1}B$. The plot shows an excellent agreement between the graphs for large values of s .

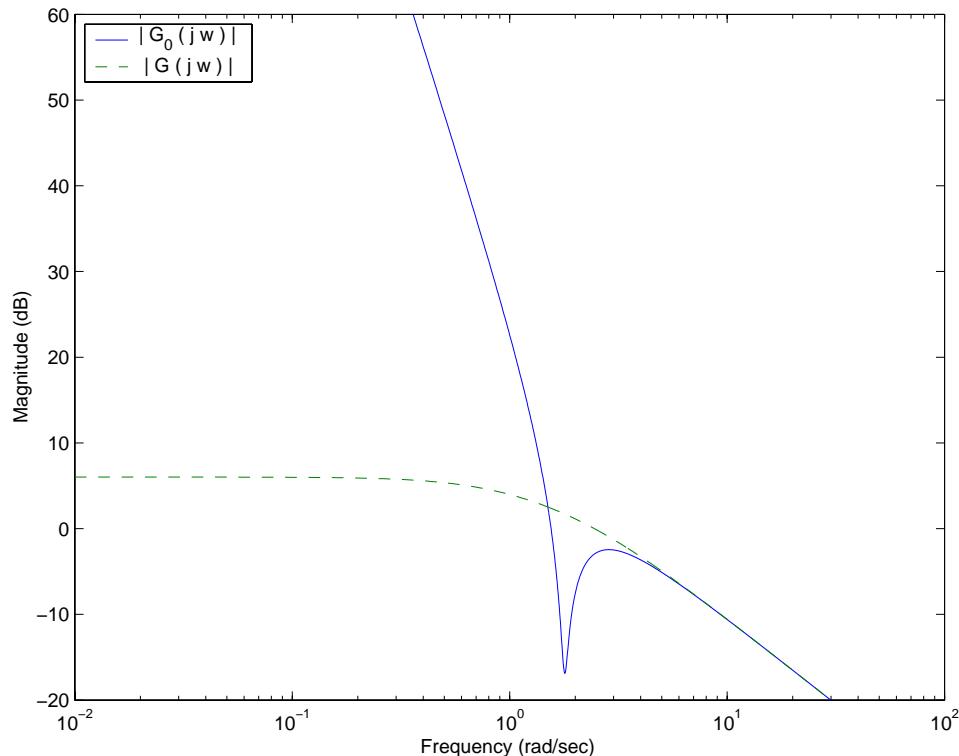


Figure 9.2: Comparison of Transfer Functions.

MATCONTROL NOTE: Algorithm 9.3.2 has been implemented in MATCONTROL function **minremsvd**.

9.4 Subspace Identification Algorithms

In this section we consider the problem of identifying the system matrices of an unknown system, **given a large number of input and output measurements. The problem is important in practical applications because it avoids computations of Markov parameters.**

We state two SVD-based subspace algorithms, one for the deterministic case and another for the stochastic case. First, we consider the deterministic case.

Specifically, the deterministic identification problem is:

Given a large number of input and output measurements, u_k and y_k , respectively of the unknown system:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned}$$

determine the order n of the unknown system and the system matrices $\{A, B, C, D\}$ up to within a similarity transformation; $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{r \times n}$ and $D \in \mathbb{R}^{r \times m}$.

9.4.1 A Subspace Deterministic Model Identification Algorithm

The algorithm has two major steps:

First, a state vector sequence is constructed as the intersection of the row spaces of two block Hankel matrices, constructed from the input/output data.

Second, the system matrices A, B, C and D are obtained from the least-squares solution of a set of linear equations.

There exists different ways to compute the intersection (see Van Overschee and De Moor (1996) for details and references). One way, presented in Moonen et al. (1989) is via the SVD of a concatenated Hankel matrix composed of two Hankel matrices defined by the input and output data, as follows:

$$H_{k|k+i} = \begin{pmatrix} Y_{k|k+i} \\ U_{k|k+i} \end{pmatrix}, \quad H_{k+1|k+2i} = \begin{pmatrix} Y_{k|k+2i} \\ U_{k|k+2i} \end{pmatrix}$$

where

$$Y_{k|k+i} = \begin{pmatrix} y_k & y_{k+1} & \cdots & y_{k+j-1} \\ y_{k+1} & y_{k+2} & \cdots & y_{k+j} \\ y_{k+2} & y_{k+3} & \cdots & y_{k+j+1} \\ \vdots & \vdots & & \vdots \\ y_{k+i-1} & y_{k+i} & \cdots & y_{k+j+i-2} \end{pmatrix}$$

$$\text{and } Y_{k|k+2i} = \begin{pmatrix} y_{k+i} & y_{k+i+1} & \cdots & y_{k+i+j-1} \\ y_{k+i+1} & y_{k+i+2} & \cdots & y_{k+i+j} \\ \vdots & \vdots & \vdots & \vdots \\ y_{k+2i-1} & y_{k+2i} & \cdots & y_{k+2i+j-2} \end{pmatrix}$$

The matrices $U_{k|k+i}$ and $U_{k|k+2i}$ are similarly defined from the input-data. Let $(X = (x_k, x_{k+1}, \dots, x_{k+j-1}))$.

The following assumptions are made:

- (I) $\text{rank}(X) = n$ (n is the minimal system order).
- (II) $\text{span}_{\text{row}}(X) \cap \text{span}_{\text{row}}(U_{k|k+i}) = \emptyset$
- (III) $\text{rank}(U_{k|k+i}) = \text{Number of rows in } U_{k|k+i}$

Theorem 9.4.1 Let the SVD of $H = \begin{pmatrix} H_{k|k+i} \\ H_{k+1|k+2i} \end{pmatrix}$ be $H = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \begin{pmatrix} S_{11} & 0 \\ 0 & 0 \end{pmatrix} V^T$. Then the state vector sequence $X_2 = (x_{k+i}, x_{k+i+1}, \dots, x_{k+i+j-1})$ is given by

$$X_2 = U_q^T U_{12}^T H_{k|k+i},$$

where U_q is defined by the SVD of $U_{12}^T U_{11} S_{11}$:

$$U_{12}^T U_{11} S_{11} = \begin{pmatrix} U_q & U_q^\perp \end{pmatrix} \begin{pmatrix} S_q & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_q^T \\ V_q^{\perp T} \end{pmatrix}$$

Proof: It can be shown [Exercise 12] that

$$\text{span}_{\text{row}}(X_2) = \text{span}_{\text{row}}(H_{k|k+i}) \cap \text{span}_{\text{row}}(H_{k+1|k+2i}).$$

Thus X_2 can be realized as a basis for the row space of $U_{12}^T H_{k|k+i}$. Then taking the SVD of $U_{12}^T H_{k|k+i}$, we have

$$\begin{aligned} U_{12}^T H_{k|k+i} &= U_{12}^T (U_{11} \ U_{12}) \begin{pmatrix} S_{11} & 0 \\ 0 & 0 \end{pmatrix} V^T \\ &= (U_{12}^T U_{11} S_{11} \ 0) V^T \\ &= (U_q S_q V_q^T \ 0) V^T \\ &= U_q (S_q \ 0) (V V_q)^T \end{aligned}$$

Noting that $U_q^T U_q = I_{n \times n}$, we obtain from above

$$U_q^T U_{12}^T H_{k|k+i} = (S_q \ 0) (V V_q)^T,$$

which is a basis for the row space of $U_{12}^T H_{k|k+i}$ and therefore is a realization of X_2 .

■

Once X_2 is determined, the system matrices A, B, C and D are identified by solving (**in the least-squares sense**) the following overdetermined set of linear equations

$$\begin{pmatrix} x_{k+i+1} & \cdots & \cdots & x_{k+i+j-1} \\ y_{k+i} & \cdots & \cdots & y_{k+i+j-2} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} x_{k+i} & \cdots & x_{k+i+j-2} \\ u_{k+i} & \cdots & u_{k+i+j-2} \end{pmatrix}.$$

It is, however, shown in the paper that the state vector sequence X_2 does not need to be explicitly computed. The system matrices A, B, C and D may be identified by making use of the already computed SVD of H . The above set of equations may then be replaced by an equivalent reduced set of equations (see **Algorithm 9.4.1** below).

This way of determining A, B, C and D is computationally more efficient.

To do this, it is useful to redefine the matrices $H_{k|k+i}$ and $H_{k+i|k+2i}$ as follows:

$$H_{k|k+i} = \begin{pmatrix} u_k & u_{k+1} & \cdots & u_{k+j-1} \\ y_k & y_{k+1} & \cdots & y_{k+j-1} \\ u_{k+1} & u_{k+2} & \cdots & u_{k+j} \\ y_{k+1} & y_{k+2} & \cdots & y_{k+j} \\ \vdots & \vdots & & \vdots \\ u_{k+i-1} & u_{k+i} & \cdots & u_{k+i+j-2} \\ y_{k+i-1} & y_{k+i} & \cdots & y_{k+i+j-2} \end{pmatrix}$$

$$H_{k+i|k+2i} = \begin{pmatrix} u_{k+i} & u_{k+i+1} & \cdots & u_{k+i+j-1} \\ y_{k+i} & y_{k+i+1} & \cdots & y_{k+i+j-1} \\ u_{k+i+1} & u_{k+i+2} & \cdots & u_{k+i+j} \\ y_{k+i+1} & y_{k+i+2} & \cdots & y_{k+i+j} \\ \vdots & \vdots & & \vdots \\ u_{k+2i-1} & u_{k+2i} & \cdots & u_{k+2i+j-2} \\ y_{k+2i-1} & y_{k+2i} & \cdots & y_{k+2i+j-2} \end{pmatrix}$$

The above Theorem still remains valid.

The following notation will be needed to state the algorithm.

$M(p : q, l : s)$ is the submatrix of M at the intersection of rows $p, p + 1, \dots, q$ and columns $l, l + 1, \dots, s$; $M(:, l : s)$ is the submatrix of M containing columns $l, l + 1, \dots, s$ and $M(p : q, :)$ is the submatrix of M containing rows $p, p + 1, \dots, q$.

Algorithm 9.4.1 (A Deterministic Subspace Identification Algorithm).

Inputs: The input and output sequence $\{u_k\}$ and $\{y_k\}$, respectively. The integers $i \geq n$, where n is the order of the system to be identified and j .

Outputs: The identified system matrices A, B, C and D .

Assumptions:

1. The system is observable.
2. The integers i and j are sufficiently large, and in particular $j >> \max(mi, ri)$, where m and r are the number of inputs and outputs.

Step 1. Calculate U and S from the SVD of H , where $H = \begin{pmatrix} H_{k|k+i} \\ H_{k+1|k+2i} \end{pmatrix}$:

$$H = USV^T = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \begin{pmatrix} S_{11} & 0 \\ 0 & 0 \end{pmatrix} V^T$$

(Note that the dimensions of U_{11}, U_{12} and S_{11} are, respectively, $(mi + ri) \times (2mi + n); (mi + ri) \times (2ri - n);$ and $(2mi + n) \times (2mi + n)$).

Step 2. Calculate the SVD of $U_{12}^T U_{11} S_{11}$:

$$U_{12}^T U_{11} S_{11} = (U_q, U_q^\perp) \begin{pmatrix} S_q & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_q^T \\ V_q^{\perp T} \end{pmatrix}$$

Step 3. Solve the following set of linear equations for A, B, C and D (in the least-squares sense):

$$\begin{pmatrix} U_q^T U_{12}^T U (mi + ri + 1 : (m+r)(i+1), :) S \\ U (mi + ri + m + 1 : (m+r)(i+1), :) S \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} U_q^T U_{12}^T U (1 : mi + ri, :) S \\ U (mi + ri + 1 : mi + ri + m, :) S \end{pmatrix}$$

Remarks: It is to be noted that the system matrices are determined from U and S only; the larger matrix V is never used in the computations. Since the matrix H whose SVD to be computed could be very large in practice, computing U and S only, without computing the full SVD of H , will be certainly very useful in practice. **Also, as stated before, the state vector sequence X_2 is not explicitly computed.**

There exists also an on-line version of the above algorithm. See Moonen et al. (1989) for details.

Example 9.4.1 Consider the following input-output data

$$u = \begin{pmatrix} 0.09130 & 0.1310 & 0.6275 & 0.1301 & -0.2206 & 0.1984 & 0.4081 & -0.0175 \\ 0.2766 & 0.7047 & 0.9173 & 0.9564 & 0.6631 & 0.7419 & 0.7479 & 1.2133 \\ 1.2427 & 1.2942 & 1.3092 & 1.1574 & 1.5600 & 1.0913 & 0.7765 \end{pmatrix}^T$$

$$y = \begin{pmatrix} 0.6197 & -0.4824 & 0.3221 & 0.2874 & -0.4582 & -0.1729 & 0.3162 & 0.0946 \\ -0.3497 & 0.3925 & 0.2446 & 0.2815 & 0.05621 & -0.2201 & 0.1397 & -0.0880 \\ 0.5250 & -0.1021 & 0.2294 & -0.0616 & -0.0706 & 0.3982 & -0.5695 \end{pmatrix}^T$$

generated from the discrete-time system:

$$\begin{aligned} x_{k+1} &= \begin{pmatrix} -0.2 & 0.3 \\ 1 & 0 \end{pmatrix} x_k + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_k \\ y_k &= (1, -1) x_k. \end{aligned}$$

Step 1. $S = \text{diag}(9.1719, 1.9793, 1.8031, 1.6608, 1.4509, 1.3426, 1.2796, 1.0657, 0.5012, 0.4554, 5.1287 \times 10^{-16}, 3.5667 \times 10^{-16}, 2.2847 \times 10^{-16}, 1.3846 \times 10^{-16}, 9.8100 \times 10^{-17}, 1.0412 \times 10^{-18})$

Step 2.

$$U_{12} = \begin{pmatrix} 0.4392 & -0.0372 & -0.1039 & 0.2139 & -0.0297 & -0.3324 \\ -0.1318 & 0.01116 & 0.0312 & -0.0642 & 0.0090 & 0.0997 \\ -0.3277 & 0.3783 & 0.1880 & 0.2071 & 0.3299 & 0.2790 \\ 0.0544 & -0.1098 & -0.0460 & -0.0836 & -0.0960 & -0.0505 \\ -0.2282 & -0.4853 & -0.2806 & 0.1674 & -0.3377 & 0.1086 \\ 0.4965 & 0.0743 & -0.0282 & 0.1215 & 0.0416 & -0.3597 \\ 0.4062 & -0.0910 & 0.3218 & -0.3029 & 0.1580 & 0.3608 \\ 0.0012 & 0.3828 & 0.0071 & 0.4531 & 0.2565 & -0.1672 \end{pmatrix}$$

$$U_{11} = \begin{pmatrix} 0.2417 & -0.2139 & 0.2202 & 0.3049 & -0.4614 & -0.1558 & 0.0638 & 0.0142 & -0.2980 & -0.2488 \\ 0.0211 & 0.3581 & 0.1578 & 0.3546 & -0.4804 & -0.0503 & -0.1130 & 0.2833 & 0.1475 & 0.5831 \\ 0.2768 & -0.3634 & -0.2593 & -0.0824 & -0.3694 & 0.2078 & 0.0116 & -0.0870 & -0.1208 & 0.0008 \\ 0.0383 & -0.5002 & 0.1157 & -0.0384 & -0.1351 & -0.0780 & 0.2107 & -0.2894 & 0.7057 & 0.1893 \\ 0.3109 & 0.0687 & -0.5137 & 0.1861 & -0.0370 & 0.0372 & 0.2759 & -0.0140 & -0.0267 & -0.0101 \\ 0.0338 & 0.0580 & -0.4635 & -0.2733 & -0.0251 & 0.3664 & -0.1282 & 0.0417 & 0.0803 & 0.3867 \\ 0.3402 & 0.0787 & -0.0426 & 0.4406 & 0.2193 & 0.2340 & -0.0493 & -0.2103 & 0.0708 & -0.0626 \\ 0.0389 & 0.2704 & -0.1147 & 0.3117 & 0.2969 & -0.2668 & 0.3531 & -0.0220 & 0.2897 & -0.0315 \end{pmatrix}$$

$$S_q = \text{diag}(1.94468, 0.624567)$$

Step 3.

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \left(\begin{array}{cc|c} 0.2635 & 0.1752 & -0.4644 \\ 1.0153 & -0.4635 & 0.5503 \\ \hline -0.1780 & 1.6527 & 1.4416 \times 10^{-16} \end{array} \right)$$

Verification: The first ten Markov parameters (denoted by $H_i, i = 1, \dots, 10$) of the original system and those of the identified system (denoted by $H'_i, i = 1, \dots, 10$) are given below:

$$\begin{aligned} H_1 &= 1, H'_1 = 0.9922 \\ H_2 &= -1.2, H'_2 = -1.1962 \\ H_3 &= 0.5400, H'_3 = 0.5369 \\ &\vdots \\ H_{10} &= -0.0343, H'_{10} = -0.0341. \end{aligned}$$

9.4.2 A Stochastic Subspace Model Identification Algorithm

We now consider the stochastic case:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + Du_k + v_k \end{aligned}$$

where $v_k \in \mathbb{R}^{r \times 1}$ and $w_k \in \mathbb{R}^{n \times 1}$ are unobserved vector signals; v_k is the measurement noise and w_k is the process noise. It is assumed that

$$E \left[\begin{pmatrix} w_p \\ v_p \end{pmatrix} (w_q^T \ v_q^T) \right] = \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \delta_{pq} \geq 0,$$

where the matrices Q, R and S are covariance matrices of the noise sequences w_k and v_k . The problem is now to determine the system matrices A, B, C and D up to within a similarity transformation and also the covariance matrices Q, S and R ; given a large number of input and output data u_k and y_k , respectively.

We state a subspace algorithm for the above problem taken from the recent paper of DeMoor, Van Overschee and Favoreel (1999). The algorithm, as in the deterministic case, determines the system matrices by first finding an estimate \tilde{X}_f of the state sequence \tilde{X} from the measurement data.

The sequence \tilde{X}_f is determined by using certain oblique projections.

Define the input Hankel matrix $U_{i|l}$ from the input data as:

$$U_{k|l} = \begin{pmatrix} u_k & u_{k+1} & \cdots & u_{k+j-1} \\ u_{k+1} & u_{k+2} & \cdots & u_{k+j} \\ \vdots & & & \\ u_l & u_{l+1} & \cdots & u_{l+j-1} \end{pmatrix}$$

Similarly, define the output Hankel matrix $Y_{k|l}$ from the output data.

The matrices A, B, C, D are then determined by solving the least-squares problem:

$$\min_{A,B,C,D} \| \begin{pmatrix} \tilde{X}_{i+1} \\ Y_{i|i} \end{pmatrix} - \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \tilde{X}_i \\ U_{i|i} \end{pmatrix} \|_F^2$$

Once the system matrices are obtained by solving above least-squares problem, the noise covariances Q, S and R can be estimated from the residuals (see **Algorithm 9.4.2 below for details**).

The algorithm, in particular, can be used to solve the deterministic problem. Thus, it can be called a **combined deterministic stochastic identification algorithm**.

Definition 9.4.1 *The oblique projection of $A \in \mathbb{R}^{p \times j}$ along the row space of $B \in \mathbb{R}^{q \times j}$ on the row space of $C \in \mathbb{R}^{r \times j}$, denoted by $A/B\mathbf{C}$ is defined as:*

$$A/B\mathbf{C} = A(C^T \ B^T) \left[\begin{pmatrix} CCT & CB^T \\ BCT & BB^T \end{pmatrix}^\dagger \right]_{\text{first } r \text{ columns}} C,$$

where \dagger denotes the Moore-Penrose pseudo-inverse of the matrix.

For convenience, following the notations of the above paper, we write

$$U_p = U_{0|i-1}, \quad U_f = U_{i|2i-1},$$

$$Y_p = Y_{0|i-1}, \quad Y_f = Y_{i|2i-1},$$

where the subscript p and f denote, respectively, the past and the future. The matrix containing the past inputs U_p and outputs Y_p will be called W_p :

$$W_p = \begin{pmatrix} Y_p \\ U_p \end{pmatrix}$$

The matrices $W_{0|i-1}$ and $W_{0|i}$ are defined in the same way as $U_{0|i-1}$ and $U_{0|i}$ from Y_p and W_p . The following assumptions are made:

- (i) The input u_k is uncorrelated with the noise w_k and v_k .
- (ii) The input covariance matrix $\frac{1}{j}(U_{0|2i-1}U_{0|2i-1}^T)$ is of full rank, that is the rank is $2mi$ (the sequence u_k is then called **persistently exciting** of order $2i$).
- (iii) The number of available measurements is sufficiently large, so that $j \rightarrow \infty$.
- (iv) The noise w_k and v_k are not identically zero.

Algorithm 9.4.2 (A Subspace Stochastic Identification Algorithm)

Inputs: The input and output sequences $\{u_k\}$, and $\{y_k\}$.

Outputs: The order of the system and the system matrices A, B, C, D .

Assumptions: As above.

Step 1. Find the oblique projections:

$$\mathcal{O}_i = Y_{i|2i-1}/U_{i|2i-1} W_{0|i-1}, \quad \mathcal{O}_{i+1} = Y_{i+1|2i-1}/U_{i+1|2i-1} W_{0|i}.$$

Step 2. Compute the SVD of the oblique projection:

$$\mathcal{O}_i = USV^T = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} \begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix},$$

(The order n of the system is equal to the order of S_1).

Step 3. Define Γ_i and Γ_{i-1} as:

$$\Gamma_i = U_1 S_1^{1/2}, \quad \Gamma_{i-1} = \underline{\Gamma_i},$$

where $\underline{\Gamma_i}$ is Γ_i without the last block row.

Step 4. Determine the state sequences:

$$\tilde{X}_i = S_1^{\frac{1}{2}} V_1^T, \quad \tilde{X}_{i+1} = \Gamma_{i-1}^\dagger \mathcal{O}_{i+1}.$$

Step 5. Solve the following linear equations (**in the least-squares sense**) for A, B, C, D and the residuals ρ_w and ρ_v :

$$\begin{pmatrix} \tilde{X}_{i+1} \\ Y_{i|i} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \tilde{X}_i \\ U_{i|i} \end{pmatrix} + \begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix}$$

Step 6. Determine the noise covariances Q, S and R from the residuals as:

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix}_i = \frac{1}{j} \left[\begin{pmatrix} \rho_w \\ \rho_v \end{pmatrix} (\rho_w^T \ \rho_v^T) \right].$$

where the index i denotes a “bias” induced for finite i , which vanishes as $i \rightarrow \infty$. ■

Implementational Remarks: In practical implementation, Step 4 should not be computed as above, because explicit computation of the latter matrix V is time consuming.

In fact, in a good software, the oblique projections in Step 1 are computed using a fast structure preserving QR factorization method and SVD in Step 2 is applied to only a part of the R -factor from the QR factorization.

For details of the proofs and practical implementations of these and other related subspace algorithms for system identification and an account of the extensive up-to-date literature (including the software on identification) on the subject, the readers are referred to the book by Van Overschee and DeMoor (1996) and the recent review paper by DeMoor, Van Overschee and Favoreel (1999).

MATLAB-Note: M -files implementing Algorithm 9.4.2 (and others) come with the book by Van Overschee and DeMoor (1996) and can also be obtained from

<ftp://www.esat.kuleuven.ac.be/pub/SISTA/vanoverschee/book/subfun/>

9.4.3 Continuous-time System Identification

Subspace system identification algorithms, analogous to Algorithm 9.4.1 and Algorithm 9.4.2, can also be developed for a continuous time system:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t). \end{aligned}$$

However, the input and output matrices have to be defined differently and they need computations of derivatives. Thus, define

$$U_{0|i-1}^c = \begin{pmatrix} u(t_0) & u(t_1) & \dots & u(t_{j-1}) \\ u^{(1)}(t_0) & u^{(1)}(t_1) & \dots & u^{(1)}(t_{j-1}) \\ \dots & \dots & \dots & \dots \\ u^{(i-1)}(t_0) & u^{(i-1)}(t_1) & \dots & u^{(i-1)}(t_{j-1}) \end{pmatrix}$$

where $u^{(p)}(t)$ denotes the p -th derivative of $u(t)$, and ‘‘c’’ stands for ‘‘**continuous**’’.

The matrices $Y_{0|i-1}^c, U_{i|2i-1}^c$, and X_i^c are similarly defined.

The continuous-time system identification problem can be stated as follows:

Given input and output measurements $u(t), y(t), t = t_0, t_1, \dots, t_{j-1}$ and the estimates of the derivatives $u^{(p)}(t)$ and $y^{(p)}(t)$ up to order $2i - 1$, of the above unknown system, find the system matrices A, B, C, D , of the above continuous-time system up to within a similarity transformation.

9.4.4 Frequency-Domain Identification

The problem we consider here is the one of identifying a **continuous-time model** given a set of frequency responses. The problem can also be solved for a discrete-time system. For frequency-domain identification of discrete-time systems, see McKelvey (1994a, 1994b, 1994c). Specifically, the frequency-domain identification problem for a continuous-time system is stated as follows:

Given N frequency domain frequency responses $G(j\omega_k)$, measured at frequencies ω_k (not necessarily distinct), $k = 1, 2, \dots, N$, find the system matrices A, B, C and D .

One indirect approach for solving the problem is to estimate the Markov parameters via matrix-fraction descriptions of the frequency responses $G(j\omega_k)$ and then apply any of the Markov parameters based time-domain algorithms described in Section 9.3. (See **Exercise 10**).

We will, however, not discuss this here. For details, the readers are referred to the book by Juang (1994). Rather, we state here a direct **subspace identification algorithm** from the paper of DeMoor, Van Overschee and Favoreel (1999).

Let $\alpha > n$ be a user supplied index. Let $Re(M)$ and $Im(M)$ denote, respectively, the real and imaginary parts of a complex matrix M . Define the following matrices from the given frequency responses:

$$\mathcal{H} = (Re(\mathcal{H}^c), Im(\mathcal{H}^c))$$

$$\mathcal{I} = (Re(\mathcal{I}^c), Im(\mathcal{I}^c)),$$

where

$$\mathcal{H}^c = \begin{pmatrix} G(j\omega_1) & G(j\omega_2) & \cdots & G(j\omega_N) \\ (j\omega_1)G(j\omega_1) & (j\omega_2)G(j\omega_2) & \cdots & (j\omega_N)G(j\omega_N) \\ \vdots & \vdots & & \vdots \\ (j\omega_1)^{\alpha-1}G(j\omega_1) & (j\omega_2)^{\alpha-1}G(j\omega_2) & \cdots & (j\omega_N)^{\alpha-1}G(j\omega_N) \end{pmatrix}$$

and

$$\mathcal{I}^c = \begin{pmatrix} I_m & I_m & \cdots & I_m \\ (j\omega_1)I_m & (j\omega_2)I_m & \cdots & (j\omega_N)I_m \\ \vdots & \vdots & & \vdots \\ (j\omega_1)^{\alpha-1}I_m & (j\omega_2)^{\alpha-1}I_m & \cdots & (j\omega_N)^{\alpha-1}I_m \end{pmatrix}$$

Algorithm 9.4.3 Continuous-time Frequency-Domain Subspace Identification Algorithm

Inputs: The set of measured frequencies $G(j\omega_1), G(j\omega_2), \dots, G(j\omega_N)$, an integer α and a weighting matrix W .

Outputs: The system matrices A, B, C , and D .

Step 1. Find the orthogonal projection of the row space of \mathcal{H} into the row space of \mathcal{I}^\perp :

$$O_\alpha = \mathcal{H} - \mathcal{H}\mathcal{I}^\dagger\mathcal{I}$$

Step 2. Compute the SVD of WO_α :

$$WO_\alpha = USV^T = (U_1, U_2) \begin{pmatrix} S_1 & 0 \\ 0 & S_0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix},$$

where W is a weighting matrix.

Step 3. Determine $\Gamma_\alpha = W^{-1}U_1S_1^{\frac{1}{2}}$.

Step 4. Determine A and C as follows:

C = the first r rows of Γ_α

$A = \underline{\Gamma}_\alpha^\dagger \bar{\Gamma}_\alpha$,

where $\bar{\Gamma}_\alpha$ and $\underline{\Gamma}_\alpha$ denote Γ_α without the first and last r rows.

Step 5. Determine B and D via the least-squares solution of the linear systems of equations:

$$\begin{pmatrix} Re(L) \\ Im(L) \end{pmatrix} = \begin{pmatrix} Re(M) \\ Im(M) \end{pmatrix} \begin{pmatrix} B \\ D \end{pmatrix},$$

where L and M are given by:

$$L = \begin{pmatrix} G(j\omega_1) \\ \vdots \\ G(j\omega_N) \end{pmatrix} \text{ and } M = \begin{pmatrix} C(j\omega_1 - A)^{-1} & I_r \\ \vdots & \vdots \\ C(j\omega_N - A)^{-1} & I_r \end{pmatrix}$$

(Note that L and M are, respectively, of order $rN \times m$ and $rN \times (n+r)$).

Remarks: 1. The choice of the weighting matrix W is very important. If W is chosen appropriately, then the results are “unbiased”; otherwise, they will be “biased.” For details of how the weighting should be chosen, the readers are referred to the paper by Van Overschee and De Moor (1996).

2. The algorithm works well when n and i are small.

However, when i grows larger, the block Hankel matrices \mathcal{H} and \mathcal{I} became very highly ill-conditioned. The paper of Van Overschee and De Moore (1996) contains a more numerically effective algorithm.

9.5 Some Selected Software

9.5.1 MATLAB CONTROL SYSTEM TOOLBOX

State-space models

`minreal` - Minimal realization and pole/zero cancellation.

`augstate` - Augment output by appending states.

9.5.2 MATCONTROL

MINRESVD - Finding minimal realization using singular value decomposition of Hankel matrix of Markov parameters (**Algorithm 9.3.1**)

MINREMSVD - Finding minimal realization using singular value decomposition of Hankel matrix of lower order (**Algorithm 9.3.2**)

9.5.3 CSP-ANM

Model identification

- The system identification from its impulse responses is performed by `ImpulseResponseIdentify` [*response*].
- The system identification from its frequency responses is performed by `FrequencyResponseIdentify` [*response*].
- The system identification directly from input-output data is performed by `OutputResponseIdentify` [*u, y*].

9.5.4 SLICOT

Identification

IB - Subspace Identification

Time Invariant State-space Systems

IB01AD Input-output data preprocessing and finding the system order

IB01BD Estimating the system matrices, covariances, and Kalman gain

IB01CD Estimating the initial state and the system matrices B and D

TF - Time Response

TF01QD Markov parameters of a system from transfer function matrix

TF01RD Markov parameters of a system from state-space representation

In addition to the above mentioned software, the following toolboxes, especially designed for system identification are available.

- **MATLAB System Identification Toolbox**, developed by Prof. Lennart Ljung. (Web-site: <http://www.mathworks.com>)

- **ADAPTX**, developed by W. E. Larimore. (Website: <http://adaptx.com>)
- **Xmath Interactive System Identification Module**, described in the manual *X-Math Interactive System Identification Module, Part 2*, by P. VanOverschee, B. DeMoor, H. Aling, R. Kosut, and S. Boyd, Integrated Systems Inc., Santa Clara, California, USA, 1994 (website: http://www.isi.com/products/MATRIX_X/Techspec/MATRIX_X-Xmath/xm36.html, [-/MATRIX_X_XMATH/inline images/pg. 37 img.html](http://MATRIX_X_XMATH/inline images/pg. 37 img.html) and [-/MATRIX_X-XMath/inlineimages/pg. 38img.html](http://MATRIX_X-XMath/inlineimages/pg. 38img.html)).

For more details on these software packages, see the paper by DeMoor, Van Overschee and Favoreel (1999).

9.5.5 MATRIX_X

Purpose: Compute the minimal realization of a system.

Syntax: [SMIN, NSMIN, T]=MINIMAL (S, NS, TOL) or
[NUMMIN, DENMIN]=MINIMAL (NUM, DEN, TOL)

9.6 Summary and Review

This chapter is concerned with state-space realization and model identification.

I. Realization. Given a transfer function matrix $G(s)$, the realization problem is the problem of finding the system matrices A, B, C and D such that $G(s) = C(sI - A)^{-1}B + D$.

For a given proper rational function $G(s)$, there always exists a state-space realization. However, **such a realization is not unique**. In **Section 9.2.1** the nonuniqueness of a realization is demonstrated by computing the two realizations of the same transfer function matrix $G(s)$: **controllable and observable realizations**.

II. Minimal Realization. A realization (A, B, C, D) of $G(s)$ is a **minimal realization** if A has the smallest possible dimension. An important result on minimal realization is that **a realization is minimal if and only if (A, B) is controllable and (A, C) is observable** (**Theorem 9.2.1**).

Two minimal realizations are related by a nonsingular transforming matrix T (**Theorem 9.2.2**).

There are many methods for computing a minimal realization, given a set of **Markov parameters** $H_k = CA^{k-1}B$, $k = 1, 2, 3, \dots$, assuming that these Markov parameters are easily obtainable from a given transfer function. Most of these methods find a minimal

realization by factoring the Hankel matrix of Markov parameters:

$$M_k = \begin{pmatrix} H_1 & H_2 & \cdots & H_k \\ H_2 & H_3 & \cdots & H_{k+1} \\ \vdots & & & \\ H_k & H_{k+1} & \cdots & H_{2k-1} \end{pmatrix}.$$

Some basic properties of this Hankel matrix M_k that play an important role in the development of these algorithms are stated and proved in **Theorem 9.3.1**.

Two numerically viable SVD-based methods for computing a minimal realization are given in Sections 9.3.2 and 9.3.3 (**Algorithms 9.3.1 and 9.3.2**).

III. Time-Domain Subspace Identification.

Many times, the Markov parameters are not easily accessible. In these cases, the system matrices must be identified from a given set of input and output data.

Two subspace algorithms for system identification: **Algorithm 9.4.1** for *deterministic identification* and **Algorithm 9.4.2** for *combined deterministic and stochastic identification*, are described in **Section 9.4**.

It is assumed that the number of input and output data are very large (goes to infinity) and that the data are ergodic.

Each of these two subspace algorithms comes in two steps. The first step consists of finding (implicitly or explicitly) some estimate \tilde{X}_i of the state sequence, while in the second step, the system matrices A, B, C and D are obtained by solving an overdetermined system (in the least-squares sense) using this state sequence \tilde{X}_i .

IV. Frequency-Domain Subspace Identification.

Finally, frequency domain subspace identification is considered in **Section 9.4.4**. The problem considered there is:

Given N frequency domain responses $G(j\omega_k)$, measured at frequencies $\omega_k, k = 1, 2, \dots, N$; find the system matrices A, B, C and D .

A **continuous-time** frequency domain subspace identification algorithm (**Algorithm 9.4.3**) is described in **Section 9.4.4**.

9.7 Chapter Notes and Further Reading

Realization theory is a classical topic in system identification. Ho and Kalman (1966) first introduced the important principles and concepts of minimal realization theory. There are now well-known books and papers in this area such as Kung (1978), Ljung (1987, 1991a, 1991b), Silverman (1971), Zeiger and McEwen (1974), Dickinson, Morf and Kailath (1974), Dickinson, Kailath and Morf (1974), Juang (1994), Norton (1986), Aström and Eykhoff (1971), Eykhoff

(1974), Rissanen (1971), DeJong (1978), Brockett (1978), Datta (1980), Gragg and Lindquist (1983). These papers and books provide a good insight into the subject of system identification from Markov parameters. The paper by Gragg and Lindquist (1983) deals with partial realization problem. The subspace system identification algorithms are the input-state-output generalizations of the realization theory and these algorithms are relatively modern.

Material on subspace algorithms in this book has been taken mostly from the recent book by Van Overschee and De Moor (1996) and the recent review paper by De Moor, Van Overschee and Favoreel (1999). **Both of these two references contain an up-to-date extensive list of papers and books on realization theory and subspace identification algorithms.** Frequency-domain identification is dealt with in some depth in the book by Juang (1994) and a Newton-type algorithm for fitting transfer functions to frequency response measurements appears in Spanos and Mingori (1993).

There exists an intimate relation between subspace system identification and frequency weighted model reduction. The frequency weighted model reduction is discussed in Chapter 14 of this book. For the details of the connection between these topics, see **Chapter 5** of the book by Van Overschee and De Moor (1996).

EXERCISES

1. Prove that there always exists a state-space realization for a proper rational function.
2. Verify that the controllable realization (A, B, C, D) and the observable realization (A', B', C', D') described in Section 9.2.1 are state-space realizations of the same transfer matrix $G(s)$.
3. Give a complete proof of Theorem 9.2.2
4. Let $G(s)$ be the transfer matrix of a SISO system and let (A, b, c, d) be a state-space realization of $G(s)$:

$$G(s) = d + c(sI - A)^{-1}b = d + \frac{b(s)}{a(s)}.$$

Prove that the realization is minimal if and only if $a(s)$ and $b(s)$ are coprime.

5. (Generating the Markov Parameters)

- (a) Show that for the discrete-time system (9.3.1) with initial condition $x_0 = 0$, the Markov parameters $H_0 = D, H_i = CA^{i-1}B, i = 1, 2, \dots, l-1$ can be determined by solving the system:

$$y = SU$$

where $y = (y_0, y_1, y_2, \dots, y_{l-1})_{r \times l}$

$$S = (H_0, H_1, H_2, \dots, H_{l-1})$$

$$U = \begin{pmatrix} u_0 & u_1 & u_2 & \cdots & u_{l-1} \\ u_0 & u_1 & \cdots & u_{l-2} & \\ \ddots & & & \vdots & \\ & \ddots & & \vdots & \\ & & & u_0 & \end{pmatrix}_{ml \times l},$$

where m is the number of inputs and r is the number of outputs; the matrix U is an $ml \times l$ block upper triangular matrix.

- (b) Assume that $A^k \approx 0$ for all time steps $k \geq p$; that is, A is discrete stable, then show that the above system can be reduced to

$$y = S'U',$$

where $y = (y_0, y_1, \dots, y_{l-1})$,

$$S' = (H_0, H_1, H_2, \dots, H_p),$$

and

$$U' = \begin{pmatrix} u_0 & u_1 & u_2 & \cdots & u_p & \cdots & u_{l-1} \\ u_0 & u_1 & \cdots & u_{p-1} & \cdots & u_{l-2} & \\ \ddots & & & & & & \vdots \\ & \ddots & & & & & \vdots \\ & & \ddots & & & & \vdots \\ 0 & & & & & \ddots & u_{l-p-1} \end{pmatrix}$$

(Note that U' is of order $m(p+1) \times l$ and S' is of order $r \times m(p+1)$).

- (c) Discuss the numerical difficulties in solving the above system and work out an example to illustrate the difficulties.
- 6. Prove that the Hankel matrix M_k can be decomposed in the form (9.3.3).
- 7. Assuming that $H_k \rightarrow 0$ as $k \rightarrow \infty$, prove that the realization obtained by Algorithm 9.3.1 is discrete-stable. (**Hint:** Show that $\|S^{-\frac{1}{2}}AS^{\frac{1}{2}}\|_2$ is less than unity).
- 8. (a) Construct a discrete-time system:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned}$$

with suitable randomly generated matrices A, B and C .

- (b) Construct sufficient number of Markov parameters using the inputs $u_0 = 1, u_i = 0, i > 1$, and assuming zero initial condition.
- (c) Apply Algorithm 9.3.1 and Algorithm 9.3.2 to identify the system matrices A, B and C .
- (d) In each case plot the transfer function of the original and the identified model.

- 9. A stable system is **balanced** if both controllability and observability Grammians are equal to a diagonal matrix (**Chapter 14**).

Prove that if Algorithm 9.3.2 starts with the Hankel matrix

$$M_{\beta,\alpha} = \begin{pmatrix} H_1 & H_2 & \cdots & H_\beta \\ H_2 & H_3 & \cdots & H_{\beta+1} \\ \vdots & & & \\ H_\alpha & H_{\alpha+1} & \cdots & H_{\alpha+\beta-1} \end{pmatrix}$$

then the algorithm gives a balanced realization when the indices α and β are sufficiently large.

10. (Frequency-Domain Realization using Markov Parameters)

Consider the frequency response function $G(z_k) = C(zI - A)^{-1}B + D$; $z_k = e^{\frac{j2\pi k}{l}}$, where l is the data-length and $z_k, k = 0, 1, \dots, l$ correspond to the frequency points at $2\pi k/l\Delta t$, with Δt being the sampling time interval.

Write $G(z_k) = Q^{-1}(z_k)R(z_k)$

where

$$\begin{aligned} Q(z_k) &= I_r + Q_1 z_k^{-1} + \dots + Q_p z_k^{-p} \\ R(z_k) &= R_0 + R_1 z_k^{-1} + \dots + R_p z_k^{-p} \end{aligned}$$

are matrix polynomials and I_r is the identity matrix of order r .

- (a) Prove that knowing $G(z_k)$, the coefficient matrices of $Q(z_k)$ and $R(z_k)$ can be found by solving a least-squares problem.
- (b) How can the complex arithmetic be avoided in part (a)?
- (c) Show how to obtain the Markov-parameters from the coefficient matrices found in (a).
- (Hint: $(\sum_{i=0}^p Q_i z^{-i})(\sum_{i=0}^\infty H_i z^{-i}) = \sum_{i=0}^p R_i z^{-i}$)
- (d) Derive an algorithm for frequency-domain realization similar to Algorithm 9.3.2 based on (a)-(c).
- (e) (Juang (1994)). Apply your algorithm to the discrete-time system model defined by the following data:

$$A = \text{diag} \left(\left(\begin{array}{cc} 0.9859 & 1.500 \\ -1.500 & 0.9859 \end{array} \right), \left(\begin{array}{cc} 0.9859 & 0.1501 \\ -0.1501 & 0.9859 \end{array} \right), \left(\begin{array}{cc} 0.6736 & 0.7257 \\ -0.725 & 0.6736 \end{array} \right), \right. \\ \left. \left(\begin{array}{cc} 0.4033 & 0.9025 \\ -0.9025 & 0.4033 \end{array} \right) \right).$$

$$B = \begin{pmatrix} -0.0407 & -0.0454 \\ -0.5384 & -0.6001 \\ 0.0746 & -0.0669 \\ 0.9867 & -0.8850 \\ 0.0164 & 0.0373 \\ 0.0376 & 0.0860 \\ -0.0460 & -0.0421 \\ -0.0711 & -0.0650 \end{pmatrix}, \quad C^T = \begin{pmatrix} 0.8570 & 1.80 \\ 0.0000 & 0.00 \\ 1.5700 & -1.2 \\ 0.0000 & 0.00 \\ 1.4030 & 1.42 \\ 0.0000 & 0.00 \\ 0.9016 & 1.78 \\ 0.0000 & 0.00 \end{pmatrix}, \quad D = \begin{pmatrix} 0. & 0. \\ 0. & 0. \end{pmatrix}.$$

by calculating two hundred frequency data points equally spaced in a data frequency ranging from 0 to 16.67 Hz, and assuming that the orders of $Q(z_k)$ and $R(z_k)$ are ten. Sketch the graphs of the true and estimated frequency response functions for the first input and first output and compare the results.

11. Consider the following discrete-time model (of a rigid body of mass m with a force f acting along the direction of the motion (Juang (1994)):

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k \\y_k &= Cx_k\end{aligned}$$

where $A = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix}$, $u_k = \frac{f}{m}$, $C = (1, 0)$.

Δt = sampling time interval

m = mass of a rigid body

f = force acting upon m along the direction of the motion.

- (a) Construct the first five Markov parameters.
 - (b) Apply Algorithm 9.3.2 to identify A, B and C .
 - (c) Show that the original and the identified models have the identical Markov parameters.
12. Using the notation of Section 9.4.1, prove that $\text{span}_{\text{row}}(X_2) = \text{span}_{\text{row}}(H_{k|k+i}) \cap \text{span}_{\text{row}}(H_{k+1|k+2i})$.
13. Modify Algorithm 9.4.2 by incorporating weighting matrices W_1 and W_2 such that W_1 is of full rank and W_2 has the property that $\text{rank}(W_{0|i-1}) = \text{rank}(W_{0|i-1}W_2)$.

References

1. K. Åström and P. Eykhoff, System identification - A survey, *Automatica*, vol. 7, pp. 123-167, 1971.
2. R. Brockett, The geometry of the partial realization problem, *Proc. IEEE Conf. Dec. Control*, pp. 1048-1052, 1978.
3. K.B. Datta, Minimal realization in companion forms, *J. Franklin Institute*, vol. 309, no. 2, pp. 103-123, 1980.
4. L. S. DeJong, Numerical aspects of recursive realization algorithms, *SIAM J. Contr. Optimiz.*, vol. 16, no. 4, pp. 646-660, 1978.
5. B. DeMoor, P. Van Overschee, and W. Favoreel, Algorithms for subspace state-space systems identification: An overview, *Applied and Computational Control, Signals, and Circuits*, Birkhauser, Boston, MA, vol. 1, pp. 247-311, 1999, (B.N. Datta, et al., Editors).
6. B. Dickinson, M. Morf, T. Kailath, A minimal realization algorithm for matrix sequences, *IEEE Trans. Automat. Control*, vol. AC-19, no. 1, pp. 31-38, 1974.
7. B. Dickinson, T. Kailath, M. Morf, Canonical matrix fraction and state space descriptions for deterministic and stochastic linear systems, *IEEE Trans. Automat. Control*, vol. AC-19, pp. 656-667, 1974.
8. P. Eykhoff, *System identification*, Wiley, London, 1974.
9. K. Glover, All optimal Hankel-norm approximation of linear multivariable systems and their L_∞ -error bounds, *Int. J. Control*, vol. 39, pp. 1115-1193, 1984.
10. B. Gopinath, On the identification of linear time-invariant systems from input-output data, *The Bell System Technical Journal*, vol. 48, no. 5, pp. 1101-1113, 1969.
11. W. Gragg and A. Lindquist, On the partial realization problem, *Lin. Alg. Appl.*, vol. 50, pp. 277-319, 1983.
12. B.L. Ho and R.E. Kalman, Efficient construction of linear state variable models from input/output functions, *Regelungstechnik*, vol. 14, pp. 545-548, 1966.
13. Jer-Nan Juang, *Applied System Identification*, Prentice Hall, Englewood Cliffs, NJ, 1994.
14. R.E. Kalman, P.L. Falb, and M.A. Arbib, *Topics in Mathematical System Theory*, McGraw Hill, New York, 1969.
15. S.Y. Kung, A new identification method and model reduction algorithm via singular value decomposition, 12th *Asilomar Conf. on Circuits, Systems and Comp.*, pp. 705-714, Asilomar, CA, 1978.

16. A. Lindquist, G. Picci, On Subspace methods identification, *Proc. Mathematical Theory of Networks and Systems*, vol. 2, pp. 315-320, 1993.
17. A. Lindquist, G. Picci, On Subspace Methods identification and stochastic model reduction, *Proc. SYSID, '94*, vol. 2, pp. 397-404, 1994.
18. L. Ljung, *System identification- Theory for the User*, Prentice Hall, Englewood Cliffs, NJ, 1987.
19. L. Ljung, *Issues in System Identification*, IEEE Control System, vol. 11, no. 1, pp. 25-29, 1991a.
20. L. Ljung, *System Identification Toolbox For Use with Matlab*, The Mathworks Inc., MA, USA, 1991b.
21. T. McKelvey, *On State-Space Models in System Identification*, Thesis no. 447, Department of Electrical Engineering, Linköping University, Sweden, 1994a.
22. T. McKelvey, An efficient frequency domain state-space identification algorithm, *Proc. 33rd IEEE Conference Dec. and Control*, pp. 3359-3364, 1994b.
23. T. McKelvey, *SSID-A MATLAB Toolbox for Multivariable State-Space Model Identification*, Dept. of Electrical Engineering, Linköping University, Linköping Sweden, 1994c.
24. M. Moonen, B. DeMoor, L. Vandenberghe, J. Vandewalle, On and off-line identification of linear state space models, *Int. J. Control*, vol. 49, no. 1, pp. 219-232, 1989.
25. J.P. Norton, *An introduction to identification*, Academic Press, London, 1986.
26. J. Rissanen, Recursive identification of linear sequences, *SIAM J. Control*, vol. 9, pp. 420-430, 1971.
27. L. Silverman, Realization of linear dynamical systems, *IEEE Trans. Automat. Control*, vol. AC-16, pp. 554-567, 1971.
28. J. T. Spanos and D. L. Mingori, Newton algorithm for filtering transfer functions to frequency response measurements, *J. Guidance, Control, and Dynamics*, vol. 16, pp. 34-39, 1993.
29. P. Van Overschee and B. DeMoor, Continuous-time frequency domain subspace system identification, *Signal Processing, Special Issue on Subspace Methods, Part II: System Identification*, vol. 52, pp. 179-194, 1996.
30. P. Van Overschee and B. DeMoor, *Subspace Identification for Linear Systems: Theory, Implementation and Applications*, Kluwer Academic Publishers, Boston/London/Dordrecht, 1996.

31. H. Zeiger, A. McEwen, Approximate linear realizations of given dimension via Ho's algorithm, *IEEE Trans. Automat. Control*, vol. 19, pp. 153, 1974.

Chapter 10

FEEDBACK STABILIZATION, EIGENVALUE ASSIGNMENT, AND OPTIMAL CONTROL

Contents

10.1 Introduction	392
10.2 State-Feedback Stabilization	393
10.2.1 Stabilizability and Controllability	394
10.2.2 Stabilization via Lyapunov Equations	396
10.3 Detectability	402
10.4 The Eigenvalue and Eigenstructure Assignment Problems	403
10.4.1 Eigenvalue Assignment by State Feedback	406
10.4.2 Eigenvalue Assignment by Output Feedback	409
10.4.3 Eigenstructure Assignment	410
10.5 The Quadratic Optimization Problems	411
10.5.1 The Continuous-time Linear Quadratic Regulator (LQR) Problem . .	412
10.5.2 The Discrete-time Linear Quadratic Regulator Problem	420
10.6 H_∞ Control Problems	421
10.6.1 Computing the H_∞ -Norm	423
10.6.2 H_∞ Control Problem: A State Feedback Case.	428
10.6.3 The H_∞ Control Problem: Output Feedback Case	430
10.7 The Complex Stability Radius and Riccati Equation	433
10.8 Some Selected Software	438
10.8.1 MATLAB CONTROL SYSTEM TOOLBOX	438
10.8.2 MATCONTROL	438
10.8.3 CSP-ANM	438

10.8.4 SLICOT	438
10.8.5 MATRIX _X	439
10.9 Summary and Review	439
10.10 Chapter Notes and Further Reading	444

Topics Covered

- State-Feedback Stabilization
- Eigenvalue Assignment
- Linear Quadratic Regulator Problems
- H_∞ Control
- Stability Radius (Revisited)

10.1 Introduction

In this chapter, we first consider the problem of stabilizing a linear control system by choosing the control vector appropriately. Mathematically, the problem is to find a feedback matrix K such that $A - BK$ is stable in the continuous-time case or is discrete-stable in the discrete-time case. Necessary and sufficient conditions are established for the existence of stabilizing feedback matrices, and Lyapunov-style methods for constructing such matrices are described in Section 10.2.

A concept dual to stabilizability, called **detectability**, is then introduced and its connection with a Lyapunov matrix equation is established.

In certain practical situations, stabilizing a system is not enough; a designer should be able to control the eigenvalues of $A - BK$ so that certain design constraints are met. This gives rise to the **eigenvalue assignment (EVA) problem** or the so called **pole placement problem**. Mathematically, the problem is to find a feedback matrix K such that $A - BK$ has a preassigned spectrum. A well-known and a very important result on the solution of this problem is: Given a real pair of matrices (A, B) and Λ , an arbitrary set of n complex numbers, closed under complex conjugation; there exists a real matrix K such that the spectrum of $A - BK$ is the set Λ if and only if (A, B) is controllable. The matrix K is unique in the single-input case.

This important result is established in **Section 10.4**. The proof of this result is constructive, and leads to several well-known formulas the most important of which is the **Ackermann formula**. However, these formulas do not yield numerically viable methods for pole placement. **Numerical methods for pole placement are presented in Chapter 11.**

Since there are no set guidelines as to where the poles (the eigenvalues) need to be placed, very often, in practice, a compromise is made in which a feedback matrix is constructed in such a

way that not only the system is stabilized, but a certain performance criterion is satisfied. This leads to the well-known **Linear Quadratic Regulator** (LQR) problem. Both continuous-time and discrete-time LQR problems are discussed in Section 10.5 of this Chapter. The solutions of the LQR problems require the solutions of certain quadratic matrix equations, called the **algebraic Riccati equations**. **Numerical methods for the algebraic Riccati equations are described in Chapter 13.**

The next topic in this Chapter is the H_∞ -control problems. Though a detailed discussion on the H_∞ -Control problems is beyond the scope of the book, some simplified versions of these problems are stated in **Section 10.6** of the present chapter. The H_∞ -control problems are concerned with stabilization of perturbed versions of a system, when certain bounds of perturbations are known. The solutions of the H_∞ -control problems also require solutions of certain algebraic Riccati equations.

The concept of **stability radius** introduced in Chapter 7 is revisited in the final section of this chapter (**Section 10.7**), where a relationship between the complex stability radius and an algebraic Riccati equation (**Theorem 10.7.3**) is established, and a bisection algorithm (**Algorithm 10.7.1**) for determining the complex stability radius is described.

Reader's Guide for Chapter 10

The readers familiar with concepts and results of state-feedback stabilizations, pole-placement, LQR design and H_∞ control can skip Sections 10.2–10.6. However, two algorithms for computing the H_∞ -norm (Algorithms 10.6.1 and 10.6.2) and material on stability radius should be of interests to most readers.

10.2 State-Feedback Stabilization

In this section, we consider the problem of stabilizing the linear system

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t).\end{aligned}\tag{10.2.1}$$

Suppose that the state vector $x(t)$ is known and let's choose

$$u(t) = v(t) - Kx(t),\tag{10.2.2}$$

where K is a constant matrix, and $v(t)$ is a reference input vector.

Then feeding this input vector $u(t)$ back into the system, we obtain the system

$$\begin{aligned}\dot{x}(t) &= (A - BK)x(t) + Bv(t) \\ y &= (C - DK)x(t) + Dv(t)\end{aligned}\tag{10.2.3}$$

The problem of stabilizing the system (10.2.1) then becomes the problem of finding K such that the system (10.2.3) becomes stable. The problem of state-feedback stabilization can, therefore, be stated as follows:

Given a pair of matrices (A, B) , find a matrix K such that $A - BK$ is stable.

Graphically, the state-feedback problem can be represented as follows:

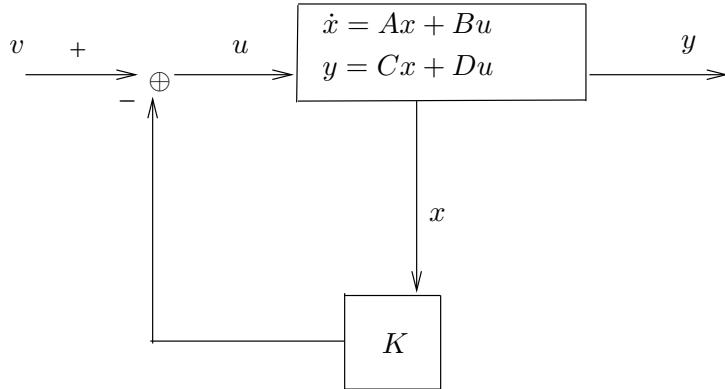


Figure 10.1: State Feedback Configuration

In the next sub-section we will investigate the conditions under which such a matrix K exists. The matrix K , when it exists, is called a **stabilizing feedback matrix**; and in this case, the pair (A, B) is called a **stabilizable pair**. The system (10.2.3) is called the **closed-loop system** and the matrix $A - BK$ is called the **closed-loop matrix**.

Analogously, for the discrete-time system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{10.2.4}$$

if there exists a matrix K such that $A - BK$ is discrete-stable that is, if it has all its eigenvalues inside the unit circle, then the pair (A, B) is called a **discrete-stabilizable pair**, and the matrix K is called a **discrete-stabilizing feedback matrix**.

In what follows, we will present simple criteria of stabilizability and algorithms for constructing stabilizing feedback matrices via Lyapunov matrix equations.

10.2.1 Stabilizability and Controllability

In this section, we describe necessary and sufficient conditions for a given pair (A, B) to be a stabilizable pair. We start with the continuous-time case.

Theorem 10.2.1 (Characterization of Continuous-time Stabilizability)

The following are equivalent:

- (i) (A, B) is stabilizable.
- (ii) $\text{rank}(A - \lambda I, B) = n$ for all $\text{Re}(\lambda) \geq 0$. In other words, the unstable modes of A are controllable.
- (iii) For all λ and $x \neq 0$ such that $x^* A = \lambda x^*$ and $\text{Re}(\lambda) \geq 0$, we have $x^* B \neq 0$.

Proof: We prove the equivalence of (i) and (ii) and leave the equivalence of (i) and (iii) as an exercise [Exercise 1].

Without any loss of generality we may assume (see Theorem 6.4.1) that the pair (A, B) is given in the form:

$$PAP^{-1} = \bar{A} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & \bar{A}_{22} \end{pmatrix}, PB = \bar{B} = \begin{pmatrix} \bar{B}_1 \\ 0 \end{pmatrix},$$

where $(\bar{A}_{11}, \bar{B}_1)$ is controllable.

Since $(\bar{A}_{11}, \bar{B}_1)$ is controllable, by the eigenvalue criterion of controllability (Theorem 6.2.1 (v)), we have $\text{rank}(\lambda I - \bar{A}_{11}, \bar{B}_1) = p$, where p is the order of \bar{A}_{11} . Therefore,

$$\text{rank}(\lambda I - \bar{A}, \bar{B}) = \text{rank} \begin{pmatrix} \lambda I - \bar{A}_{11} & -\bar{A}_{12} & \bar{B}_1 \\ 0 & \lambda I - \bar{A}_{22} & 0 \end{pmatrix} < n$$

if and only if $\text{rank}(\lambda I - \bar{A}_{22}) < n - p$, that is, if and only if λ is an eigenvalue of \bar{A}_{22} .

The proof now follows from the fact that if (A, B) is a stabilizable pair, the matrix \bar{A}_{22} must be a stable matrix. This can be seen as follows:

The stabilizability of the pair (A, B) implies the stabilizability of the pair (\bar{A}, \bar{B}) .

Since (\bar{A}, \bar{B}) is a stabilizable pair, there exists a matrix \bar{K} such that $\bar{A} - \bar{B}\bar{K}$ is stable. This means that if $\bar{K} = (\bar{K}_1, \bar{K}_2)$, then the matrix $\begin{pmatrix} \bar{A}_{11} - \bar{B}_1\bar{K}_1 & \bar{A}_{12} - \bar{B}_1\bar{K}_2 \\ 0 & \bar{A}_{22} \end{pmatrix}$ is a stable matrix, which implies that \bar{A}_{22} must be stable. ■

Corollary 10.2.1 *If the pair (A, B) is controllable, then it must be stabilizable.*

Proof: If (A, B) is controllable, then again by the eigenvalue criterion of controllability, $\text{rank}(A - \lambda I, B) = n$ for every λ . In particular, $\text{rank}(A - \lambda I, B) = n$ for every λ for which $\text{Re}(\lambda) \geq 0$. Thus, (A, B) is stabilizable. ■

The above result tells us that **the controllability implies stabilizability**.

However, the converse is not true. The stabilizability is guaranteed as long as the unstable modes are controllable.

The following simple example illustrates the fact.

$$\text{Let } A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & -3 \end{pmatrix}, b = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}.$$

(A, b) is not controllable; $\text{rank}(b, Ab, A^2b) = 2$.

However, the row vector $f^T = (-126.5, -149.5, 0)$ is such that the eigenvalues of $A - bf^T$ are $\{-10 \pm 11.4891j, -3\}$.

So, $A - bf^T$ is stable; that is, (A, b) is stabilizable.

The Discrete-Case

A theorem, analogous to Theorem 10.2.1, can be proved for the discrete-time system as well. We state the result without proof. The proof is left as an exercise [**Exercise 2**].

Theorem 10.2.2 (Characterization of Discrete-Stabilizability).

The following conditions are equivalent:

- (i) The pair (A, B) is discrete-stabilizable.
- (ii) $\text{rank}(A - \lambda I, B) = n$ for every λ such that $|\lambda| \geq 1$.
- (iii) For all λ and $x \neq 0$ such that $x^*A = \lambda x^*$ and $|\lambda| \geq 1$, we have $x^*B \neq 0$.

10.2.2 Stabilization via Lyapunov Equations

From the discussions of the previous section, it is clear that for finding a feedback stabilizing matrix K for a given pair (A, B) , we can assume that the pair (A, B) is controllable. For, if (A, B) is not controllable but stabilizable, then we can always put it in the form:

$$\begin{aligned} TAT^{-1} &= \bar{A} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & \bar{A}_{22} \end{pmatrix}, \\ TB &= \bar{B} = \begin{pmatrix} \bar{B}_1 \\ 0 \end{pmatrix} \end{aligned} \tag{10.2.5}$$

where $(\bar{A}_{11}, \bar{B}_1)$ is controllable, and \bar{A}_{22} is stable.

Once a stabilizing matrix \bar{K}_1 for the controllable pair $(\bar{A}_{11}, \bar{B}_1)$ is obtained, the stabilizing matrix K for the pair (A, B) can be obtained as

$$K = \bar{K}T,$$

where

$$\bar{K} = (\bar{K}_1, \bar{K}_2), \tag{10.2.6}$$

and \bar{K}_2 is arbitrary.

We can therefore concentrate on stabilizing a controllable pair. The following theorem shows how to stabilize a controllable pair using a Lyapunov equation.

Theorem 10.2.3 *Let (A, B) be controllable and let β be a scalar such that*

$$\beta > |\lambda_{\max}(A)|,$$

where $\lambda_{\max}(A)$ is the eigenvalue of A with the largest real part. Let K be defined by

$$K = B^T Z^{-1}, \quad (10.2.7)$$

where Z (necessarily symmetric positive definite) satisfies the Lyapunov equation

$$-(A + \beta I)Z + Z[-(A + \beta I)]^T = -2BB^T, \quad (10.2.8)$$

then $A - BK$ is stable; that is, (A, B) is stabilizable.

Proof: Since $\beta > |\lambda_{\max}(A)|$, the matrix $-(A + \beta I)$ is stable.

Also, since (A, B) is controllable, the pair $(-(A + \beta I), B)$ is controllable. Thus by Theorem 7.2.6 the Lyapunov equation (10.2.8) has a unique symmetric positive definite solution Z .

Again, the equation (10.2.8) can be written as

$$(A - BB^T Z^{-1})Z + Z(A - BB^T Z^{-1})^T = -2\beta Z.$$

From (10.2.7) we then have

$$(A - BK)Z + Z(A - BK)^T = -2\beta Z. \quad (10.2.9)$$

Since Z is symmetric positive definite, $A - BK$ is stable, by Theorem 7.2.3.

This can be seen as follows:

Let μ be an eigenvalue of $A - BK$ and y be the corresponding eigenvector.

Then multiplying the both sides of the equation (10.2.9) first by y^* to the left and then by y to the right, we have

$$2 \operatorname{Re}(\mu)y^*Zy = -2\beta y^*Zy.$$

Since Z is positive definite, $y^*Zy > 0$. Thus $\operatorname{Re}(\mu) < 0$. So, $A - BK$ is stable. ■

The above discussion leads to the following method for finding a stabilizing feedback matrix (see Armstrong (1975)).

A Lyapunov-Equation Method For Stabilization

Let (A, B) be a controllable pair. Then the following method computes a stabilizing feedback matrix K .

Step 1. Choose a number β such that $\beta > |\lambda_{\max}(A)|$, where $\lambda_{\max}(A)$ denotes the eigenvalue of A with the largest real part.

Step 2. Solve the Lyapunov equation (10.2.8) for Z :

$$-(A + \beta I)Z + Z[-(A + \beta I)]^T = -2BB^T.$$

Step 3. Obtain the stabilizing feedback matrix K

$$K = B^T Z^{-1}.$$

MATCONTROL Note: The above method has been implemented in MATCONTROL function **stablyapc**.

A Remark on Numerical Effectiveness

The Lyapunov equation in Step 2 can be highly ill-conditioned, even when the pair (A, B) is robustly controllable. In this case, the entries of the stabilizing feedback matrix K are expected to be very large, giving rise to practical difficulties in implementation. See the example below.

Example 10.2.1 (Stabilizing the motion of the Inverted Pendulum) Consider Example 5.2.5 (The problem of a cart with inverted pendulum) with the following data:

$$\begin{aligned} m &= 1\text{kg} \\ M &= 2\text{kg} \\ l &= 0.5 \text{ meters} \\ \text{and } g &= 9.18 \text{ meters per sec}^2. \end{aligned}$$

Then

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -3.6720 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 22.0320 & 0 \end{pmatrix}.$$

The eigenvalues of A are $0, 0, \pm 4.6938$. Thus, with no control input, there is an instability in the motion and the pendulum will fall. We will now stabilize the motion by using the Lyapunov-equation method with A as given above, and

$$B = \begin{pmatrix} 0 \\ 0.4 \\ 0 \\ -0.4 \end{pmatrix}.$$

Step 1. Let's choose $\beta = 5$. This will make $-(A + \beta I)$ stable.

$$\text{Step 2. } Z = \begin{pmatrix} 0.0009 & -0.0044 & -0.0018 & 0.0098 \\ -0.0044 & 0.0378 & 0.0079 & -0.0593 \\ -0.0018 & 0.0079 & 0.0054 & -0.0270 \\ 0.0098 & -0.0593 & -0.0270 & 0.1508 \end{pmatrix}$$

(The computed Z is symmetric positive definite but highly ill-conditioned).

Step 3. $K = B^T Z^{-1} = 10^3(-0.5308, -0.2423, -1.2808, -0.2923)$

Verify: The eigenvalues of $A - BK$ are $\{-5 \pm 11.2865j, -5 \pm 0.7632j\}$.

Note that the entries of K are large. The pair (A, B) is, however, robustly controllable, which is verified by the fact that the singular values of the controllability matrix are 8.9462, 8.9462, 0.3284, 0.3284.

Remark: If the pair (A, B) is not controllable, but stabilizable; then after transforming the pair (A, B) to the form (\bar{A}, \bar{B}) given by

$$TAT^{-1} = \bar{A} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & A_{22} \end{pmatrix}, \quad TB = \bar{B} = \begin{pmatrix} \bar{B}_1 \\ 0 \end{pmatrix},$$

we will apply the above method to the pair $(\bar{A}_{11}, \bar{B}_1)$ (which is controllable) to find a stabilizing feedback matrix \bar{K}_1 for the pair (\bar{A}_{11}, B_1) and then obtain K that stabilizes the pair (A, B) as

$$K = (\bar{K}_1, \bar{K}_2) T,$$

where \bar{K}_2 is arbitrarily chosen.

Example 10.2.2 Consider the uncontrollable, but the stabilizable pair (A, B) :

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}.$$

Step 1. $A = \bar{A}$, $B = \bar{B}$. So, $T = I$.

$$\bar{A}_{11} = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}, \quad \bar{A}_{22} = -3, \quad \bar{B}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Step 2. Choose $\beta_1 = 10$

The unique symmetric positive definite solution Z_1 of the Lyapunov equation

$$-(\bar{A}_{11} + \beta_1 I)Z_1 + Z_1[-(\bar{A}_{11} + \beta_1 I)]^T = -2\bar{B}_1\bar{B}_1^T$$

is

$$Z_1 = \begin{pmatrix} 0.0991 & -0.0906 \\ -0.0906 & 0.0833 \end{pmatrix}.$$

Step 3. $\bar{K}_1 = \bar{B}_1^T Z_1^{-1} = (-126.5, -149.5)$.

Step 4. Choose $\bar{K}_2 = 0$. Then $K = \bar{K} = (\bar{K}_1, \bar{K}_2) = (-126.5, -149.5, 0)$.

Verify: The eigenvalues of $A - BK$ are $-10 \pm 11.489j, -3$.

Discrete-stabilization via Lyapunov Equation

The following is a discrete-analog of Theorem 10.2.3. We state the theorem without proof. The proof is left as an exercise [**Exercise 3**].

Theorem 10.2.4 *Let the discrete-time system*

$$x_{k+1} = Ax_k + Bu_k$$

be controllable. Let $0 < \beta \leq 1$ be such that $|\lambda| \geq \beta$ for any eigenvalue λ of A . Define $K = B^T(Z + BB^T)^{-1}A$, where Z satisfies the Lyapunov equation

$$AZA^T - \beta^2 Z = 2BB^T,$$

then $A - BK$ is discrete-stable.

■

The above Theorem leads to the following Lyapunov method of discrete-stabilization. The method is due to Armstrong and Rublein (1976).

A Lyapunov Equation Method for Discrete-Stabilization

Step 1. Find a number β such that $0 < \beta < \min(1, \min_i |\lambda_i|)$, where $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of (A) .

Step 2. Solve the discrete Lyapunov equation for Z :

$$AZA^T - \beta^2 Z = 2BB^T$$

Step 3. Compute the discrete-stabilizing feedback matrix K

$$K = B^T(Z + BB^T)^{-1}A.$$

Example 10.2.3 *Consider the cohort population model from Chapter 5 (Example 5.2.7),*

with $\alpha_1 = \alpha_2 = \alpha_3 = 1, \beta_1 = \beta_2 = \beta_3 = \beta_4 = 1$, and $B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$.

Then $A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$.

The eigenvalues of A are $-1.9276, -0.7748, -0.0764 \pm 0.8147j$. The matrix A is not discrete-stable.

Step 1. Choose $\beta = 0.5$

Step 2. The solution Z to the discrete Lyapunov equation $AZAT^T - \beta^2 Z = 2BB^T$ is

$$Z = -\begin{pmatrix} -0.0398 & 0.0321 & -0.0003 & 0.0161 \\ 0.0321 & -0.1594 & 0.1294 & -0.0011 \\ -0.0003 & 0.1214 & -0.6376 & 6.5135 \\ 0.0161 & -0.0011 & 6.5135 & -2.5504 \end{pmatrix}.$$

Step 3. $K = (1.2167, 1.0342, 0.9886, 0.9696)$

Verify: The eigenvalues of $A - BK$ are $-0.0742 \pm 0.4259j$, -0.4390 , and 0.3708 .

Thus $A - BK$ is discrete-stable.

Note: If (A, B) is not a discrete-controllable pair, but is discrete-stabilizable, then we can proceed exactly in the same way as in the continuous-time case to stabilize the pair (A, B) . The following example illustrates how to do this.

MATCONTROL Note: Discrete Lyapunov stabilization method as described above has been implemented in MATCONTROL function **stablypd**.

Example 10.2.4 Let

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & -1 & 1 \\ 0 & 0 & -0.9900 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The pair (A, B) is not discrete-controllable, but is discrete-stabilizable.

Using the notations of Section 10.2.2, we have $\bar{A} = A$, $\bar{B} = B$. The eigenvalues of \bar{A} are

$$\{1.7321, -1.7321, -0.9900\}.$$

$$\bar{A}_{11} = \begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix}, \quad \bar{B}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \quad \text{The pair } (\bar{A}_{11}, \bar{B}_1) \text{ is controllable.}$$

We now apply the Lyapunov method of discrete-stabilization to the pair $(\bar{A}_{11}, \bar{B}_1)$.

Step 1. Choose $\beta = 1$.

Step 2. The solution Z_1 of the discrete Lyapunov equation: $\bar{A}_{11}Z_1\bar{A}_{11}^T - Z_1 = 2\bar{B}_1\bar{B}_1^T$ is

$$Z_1 = \begin{pmatrix} 0.5 & 0.25 \\ 0.25 & 0.25 \end{pmatrix}.$$

Step 3.

$$\bar{K}_1 = (0, 2.4000).$$

Step 4. The matrix $\bar{A}_{11} - \bar{B}_1\bar{K}_1$ is discrete-stable. To obtain \bar{K} such that $\bar{A} - \bar{B}\bar{K}$ is discrete-stable, we choose $\bar{K} = (\bar{K}_1, 0)$. The eigenvalue of $\bar{A} - \bar{B}\bar{K}$ are $0.7746, -0.7746, -0.9900$, showing that $\bar{A} - \bar{B}\bar{K}$ is discrete-stable, that is, $A - B\bar{K}$ is discrete-stable.

Remark: For an efficient implementation of the Lyapunov method for feedback stabilization using the Schur method, see Sima (1981).

10.3 Detectability

As observability is a dual concept of controllability, a concept dual to stabilizability is called **detectability**.

Definition 10.3.1 *The pair (A, C) is **detectable** if there exists a matrix L such that $A - LC$ is stable.*

By duality of Theorem 10.2.1 we can then state the following result. The proof is left as an exercise [**Exercise 8**].

Theorem 10.3.1 (Characterization of Continuous-time Detectability) *The following conditions are equivalent:*

- (i) (A, C) is detectable.
- (ii) The matrix $\begin{pmatrix} A - \lambda I \\ C \end{pmatrix}$ has full column rank for all $Re(\lambda) \geq 0$.
- (iii) For all λ and $x \neq 0$ such that $Ax = \lambda x$ and $Re(\lambda) \geq 0$, we have $Cx \neq 0$.
- (iv) (A^T, C^T) is stabilizable.

■

We have seen in Chapter 7 that the controllability and observability play important role in the existence of positive definite and semidefinite solutions of Lyapunov equations.

Similar results, therefore, should be expected involving detectability. We prove one such result in the following.

Theorem 10.3.2 (Detectability and Stability)

Let (A, C) be detectable and let the Lyapunov equation

$$XA + A^T X = -C^T C \quad (10.3.1)$$

have a positive semidefinite solution X . Then A is a stable matrix.

Proof: The proof is by contradiction. Suppose that A is unstable. Let λ be an eigenvalue of A with $Re(\lambda) \geq 0$ and x be the corresponding eigenvector. Then premultiplying the equation (10.3.1) by x^* and postmultiplying it by x , we obtain $2Re(\lambda)(x^* X x) + x^* C^T C x = 0$. Since $X \geq 0$ and $Re(\lambda) \geq 0$, we must have that $Cx = 0$. This contradicts the fact that (A, C) is detectable. ■

Discrete-Detectability

Definition 10.3.2 *The pair (A, C) is discrete-detectable if there exists a matrix L such that $A - LC$ is discrete-stable.*

Theorems analogous to Theorem 10.3.1 and 10.3.2 also hold in the discrete case. We state the discrete counterpart of Theorem 10.3.1 below and leave the proof as an exercise [**Exercise 10**].

Theorem 10.3.3 *The following are equivalent:*

- (i) (A, C) is discrete-detectable.
- (ii) $\text{Rank} \begin{pmatrix} A - \lambda I \\ C \end{pmatrix} = n$ for every λ such that $|\lambda| \geq 1$.
- (iii) For all λ and $x \neq 0$ such that $Ax = \lambda x$ and $|\lambda| \geq 1$, we have $Cx \neq 0$.
- (iv) (A^T, C^T) is discrete-stabilizable.

■

10.4 The Eigenvalue and Eigenstructure Assignment Problems

We have just seen how an unstable system can be possibly stabilized by using feedback control. However, in practical instances, stabilization alone may not be enough. The stability of the system needs to be monitored and/or the system response needs to be altered. To meet certain design constraints, a designer should be able to choose the feedback matrix such that the closed-loop system has certain transient properties defined by the eigenvalues of the system. We illustrate this with the help of a second-order system.

Consider the second-order system:

$$\ddot{x}(t) + 2\zeta\omega_n\dot{x}(t) + \omega_n^2x(t) = u(t).$$

The poles of this second-order system are of the form: $\lambda_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$. The quantity ζ is called the **damping ratio** and ω_n is called the undamped **natural frequency**. The responses of the dynamical system depends upon ζ and ω_n . *In general, for a fixed value of ω_n , the larger the value of ζ ($\zeta \geq 1$) is, smoother but slower the responses become; on the contrary, the smaller the value of ζ ($0 \leq \zeta < 1$) is, the faster but more oscillatory the response is.* The following two graphs illustrate the situations.

For the first graph, $\omega_n = 1$ and $\zeta = 3$. It takes about 8 time units to reach the steady state value 1.

For the second graph, $\omega_n = 1$ and $\zeta = 0.5$. The response is much faster as it reaches the steady state value 1 in about 3 units time. However, it does not maintain that value, it oscillates before it settles down to 1.

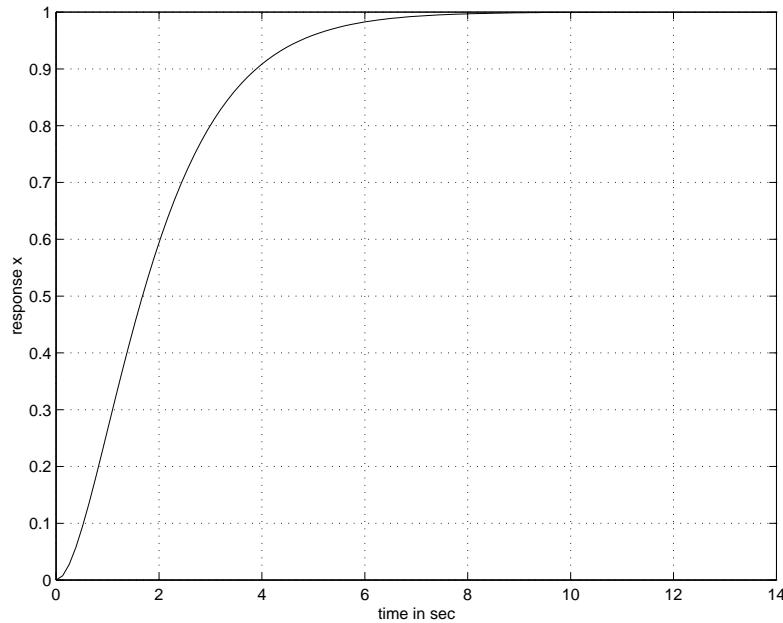


Figure 10.2 Unit Step Response when $\zeta = 3$ and $\omega_n = 1$.

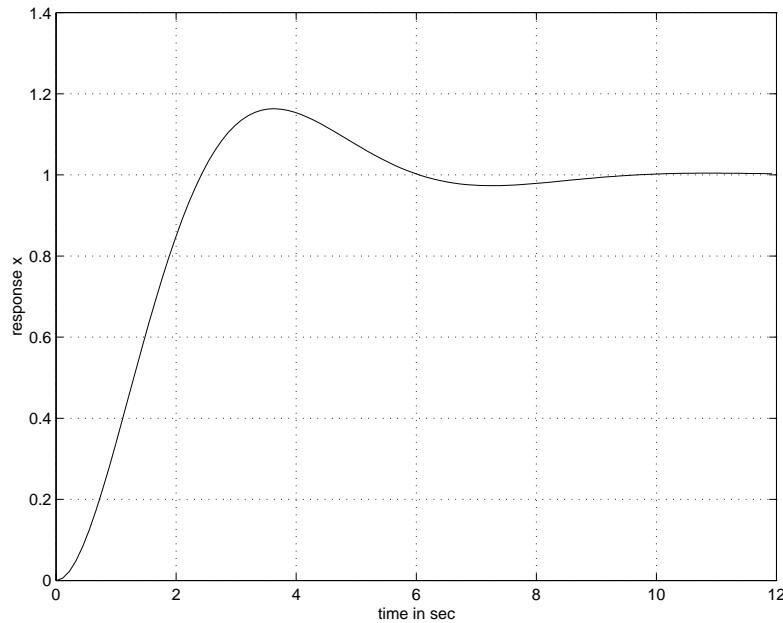


Figure 10.3 Unit Step Response when $\zeta = 0.5$ and $\omega_n = 1$.

These quantities thus need to be chosen according to a desired transient response. If the poles are close to $j\omega$ -axis in the left half s -plane, then the transient responses decay relatively slowly. On the other hand, the poles far away from the $j\omega$ -axis cause rapidly decaying time responses. Normally, “*The closed-loop poles for a system can be chosen as a desired pair of*

dominant second-order poles, with the rest of the poles selected to have real parts corresponding to sufficiently damped modes so that the system will mimic a second-order response with a reasonable control effort" (Franklin, et al. (1986)). The dominant poles are the poles that have dominant effects on the transient response behavior. As far as transient response is concerned, the poles with magnitudes of real parts at least five times greater than the dominant poles may be considered as insignificant. We give below some illustrative examples.

Case 1. Suppose that it is desired that the closed-loop system response have the minimum decay rate $\alpha > 0$, that is $\operatorname{Re}(\lambda) \leq -\alpha$ for all eigenvalues λ . Then the eigenvalues should lie in the shifted half plane as shown in Figure 10.4.

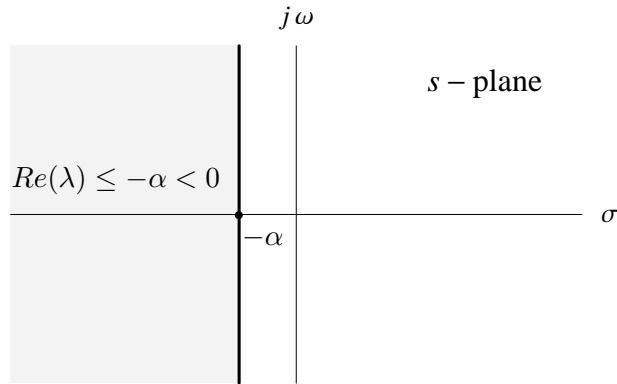


Figure 10.4: The Minimum Decay Rate α of the closed-loop system

Case 2. Suppose that it is desired that the system have the minimal damping ratio ζ_{\min} . Then the eigenvalues should lie in the sector as shown in Figure 10.5.

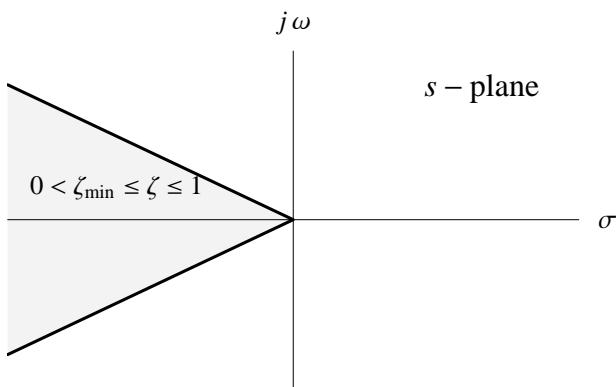


Figure 10.5: Minimal Damping Ratio ζ of the closed-loop system: The poles lie in the sector $\{\lambda \in \mathbb{C} : |\operatorname{Im}(\lambda)| \leq -\operatorname{Re}(\lambda) \sqrt{\zeta^{-2} - 1}\}$.

Case 3. Suppose that it is desired that the closed-loop system have a minimal undamped frequency ω_{\min} . Then the eigenvalues of the closed-loop matrix should lie outside of the following half of the disk: $0 < \omega_{\min} \leq \omega_n$, as shown in Figure 10.6.

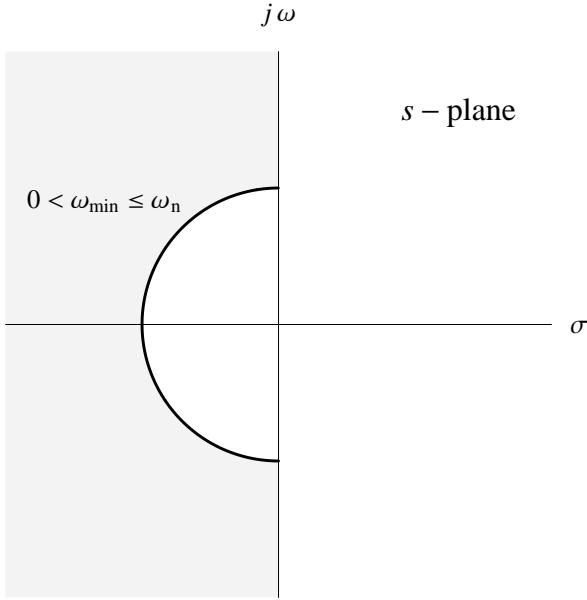


Figure 10.6: The minimal undamped frequency ω_{\min} of the closed loop system: The poles lie in the region $\{\lambda \in \mathbb{C} : |\lambda| \geq \omega_{\min}\}$.

Knowing that to obtain certain transient responses, the eigenvalues of the closed loop system should be placed in certain specified regions of the complex plane, the question arises: **where should these eigenvalues be placed?** An excellent discussion to this effect is given in the books by Friedland (Friedland (1986), 243-246) and Kailath ((1980), Chapter 3).

If the eigenvalues of the closed loop system are moved far from those of the open loop system, then from the explicit expression of the feedback vector (to be given later) in the single-input case, it is easily seen that a large feedback f will be required. From the control law

$$u = v - f^T x(t)$$

it then follows that this would require large control inputs, and there are practical limitations on how large control inputs can be.

Thus, although the eigenvalues have to be moved to stabilize a system, “**the designer should not attempt to alter the dynamic behavior of the open-loop process more than is required**” (Friedland (1986)).

10.4.1 Eigenvalue Assignment by State Feedback

The problem of assigning the eigenvalues at certain desired locations in the complex plane using the control law (10.2.2) is called the **Eigenvalue Assignment Problem by state**

feedback or in short EVA. In control theory literature, it is more commonly known as the **pole placement problem**.

Here is the precise mathematical statement of the problem.

Given $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ ($m \leq n$) and $\Lambda = \{\lambda_1, \dots, \lambda_n\}$, where Λ is closed under complex conjugation, find $K \in \mathbb{R}^{m \times n}$ such that

$$\Omega(A - BK) = \Lambda.$$

Here $\Omega(R)$ stands for the spectrum of R .

The matrix K is called the **state feedback matrix**.

The following theorem gives the conditions of existence and uniqueness of K .

Theorem 10.4.1 (The State Feedback Eigenvalue Assignment Theorem) *The EVA problem is solvable for all Λ if and only if (A, B) is controllable. The solution is unique if and only if the system is a single-input system (that is, if B is a vector). In the multi-input case, if the problem is solvable, there are infinitely many solutions.*

Proof: We first prove the **necessity**. The proof is by contradiction.

Suppose that the pair (A, B) is not controllable. Then according to the eigenvalue criteria of controllability, we have $\text{rank}(A - \lambda I, B) < n$ for some λ . Thus there exists a vector $z \neq 0$ such that $z^T(A - \lambda I) = 0, z^T B = 0$. This means that for any K , we have $z^T(A - \lambda I - BK) = 0$, which implies that λ is an eigenvalue of $A - BK$ for every K , and thus λ cannot be reassigned.

Next we prove the **sufficiency**.

Case 1. Let's consider first the **single-input case**. That is, we prove that if (A, b) is controllable, then there exists a unique vector f such that the matrix $A - bf^T$ has the desired spectrum.

Consider the (lower) controller-companion form (C, \tilde{b}) of the controllable pair (A, b) (see **Chapter 6**):

$$TAT^{-1} = C = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_1 & -a_2 & -a_3 & \cdots & -a_n \end{pmatrix} \quad (10.4.1)$$

and

$$\tilde{b} = Tb = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}. \quad (10.4.2)$$

We now show that there exists a row vector \hat{f}^T such that the closed-loop matrix $C - \tilde{b}\hat{f}^T$ has the desired spectrum.

Let the characteristic polynomial of the desired closed-loop matrix be $d(\lambda) = \lambda^n + d_n\lambda^{n-1} + \cdots + d_1$. Let $\hat{f}^T = (\hat{f}_1, \hat{f}_2, \dots, \hat{f}_n)$.

Then

$$C - \tilde{b}\hat{f}^T = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_1 - \hat{f}_1 & -a_2 - \hat{f}_2 & \cdots & & -a_n - \hat{f}_n \end{pmatrix}. \quad (10.4.3)$$

The characteristic polynomial $c'(\lambda)$ of $C - \tilde{b}\hat{f}^T$, then, is $\lambda^n + (a_n + \hat{f}_n)\lambda^{n-1} + \cdots + a_1 + \hat{f}_1$. Comparing the coefficients of $c'(\lambda)$ with those of $d(\lambda)$, we immediately have

$$\hat{f}_i = d_i - a_i, \quad i = 1, 2, \dots, n. \quad (10.4.4)$$

Thus the vector \hat{f} is completely determined by the coefficients of the characteristic polynomial of the matrix C and the coefficients of the characteristic polynomial of the desired closed-loop matrix. Once the vector \hat{f} is known, the vector f such that the original closed-loop matrix $A - bf^T$ has the desired spectrum, can now be found from the relation

$$f^T = \hat{f}^T T. \quad (10.4.5)$$

(Note that $\Omega(A - bf^T) = \Omega(TAT^{-1} - Tb\hat{f}^TT^{-1}) = \Omega(C - \tilde{b}\hat{f}^T)$).

Uniqueness: From the construction of \hat{f} , it is clear that \hat{f} is unique. We now show that the uniqueness of \hat{f} implies that of f . The proof is by contradiction.

Suppose there exists $g \neq f$ such that $\Omega(A - bg^T) = \Omega(A - bf^T)$. Then $\Omega(C - \tilde{b}\hat{f}^T) = \Omega(C - \tilde{b}\hat{g}^T)$, where $\hat{g}^T = g^T T^{-1} \neq \hat{f}^T$, which contradicts the uniqueness of the vector \hat{f} .

Case 2.

Now we turn to the **multi-input case**. Since (A, B) is controllable, there exists a matrix F and a vector g such that $(A - BF, Bg)$ is controllable (see Chen (1984), pp. 344). Thus, by Case 1, there exists a vector h such that the matrix $A - BF - Bgh^T$ has the desired spectrum. Then with $K = F + gh^T$, we have that $A - BK$ has the desired spectrum.

Uniqueness: Since the choice of the pair (F, g) is not unique, there exist infinitely many feedback matrices K in the multi-input case.

■

The Bass-Gura Formula

Note that using the expression of T from Chapter 6, the above expression for f in the single-input case can be written as [Exercise 13]:

$$f = T^T \hat{f} = [(C_M W)^T]^{-1} (d - a), \quad (10.4.6)$$

where d is the vector of the coefficients of the desired characteristic polynomial, a is the vector of the coefficients of the characteristic polynomial of A , C_M is the controllability matrix, and W is a certain Toeplitz matrix.

The above formula for f is known as the **Bass-Gura formula** (see Kailath (1980), pp. 199).

Ackermann's Formula (Ackermann (1972)).

A closely related formula for the single-input feedback vector f is the well-known Ackermann formula:

$$f = e_n^T C_M^{-1} d(A), \quad (10.4.7)$$

where C_M is the controllability matrix and $d(A)$ is the characteristic polynomial of the desired closed-loop matrix.

We also leave the derivation of Ackermann's formula as an exercise [**Exercise 14**].

Notes: We remind the readers again that, since $T = C_M^{-1}$ can be very ill-conditioned, **computing f using the constructive proof of Theorem 10.4.1 or by the Ackermann or by the Bass-Gura formula can be highly numerically unstable**. We will give some numerical examples in **Chapter 11** to demonstrate this.

The MATLAB function **acker** has implemented Ackermann's formula and **comments have been made about the numerical difficulty with this formula in the MATLAB user's manual**.

10.4.2 Eigenvalue Assignment by Output Feedback

Solving the eigenvalue assignment problem by using the feedback law (10.2.2) requires knowledge of the full state vector $x(t)$.

Unfortunately, in certain situations, the full state is not measurable or it becomes expensive to feedback each state variable when the order of the system is large.

In such situations, the feedback law using the output is more practical.

Thus if we define the output feedback law by

$$u(t) = -Ky(t), \quad y(t) = Cx(t) \quad (10.4.8)$$

we have the closed-loop system

$$\dot{x}(t) = (A - BKC)x(t).$$

The **output feedback eigenvalue assignment problem** then can be defined as follows.

Given the system (10.2.1), find a feedback matrix K such that the matrix $A - BKC$ has a preassigned set of eigenvalues.

The following is a well-known result by Kimura (1975) on the solution of the output feedback problem.

Theorem 10.4.2 (The Output Feedback Eigenvalue Assignment Theorem)

Let (A, B) be controllable and (A, C) be observable. Let $\text{rank}(B) = m$ and $\text{rank}(C) = r$. Assume that $n \leq r + m - 1$. Then an almost arbitrary set of distinct eigenvalues can be assigned by the output feedback law (10.4.8). ■

10.4.3 Eigenstructure Assignment

So far, we have considered the problem of only assigning the eigenvalues. However, **if the system transient response needs to be altered, then the problem of assigning both eigenvalues and eigenvectors needs to be considered.** This can be seen as follows. We have taken the discussion here from Andry, Shapiro and Chung (1983).

Suppose that the eigenvalues $\lambda_k, k = 1, \dots, n$ of A are distinct. Let $M = (v_1, \dots, v_n)$ be the matrix of eigenvectors, which is necessarily nonsingular.

Then every solution $x(t)$ of the system

$$\dot{x}(t) = Ax(t), x(0) = x_0,$$

representing a free response can be written as

$$x(t) = \sum_{i=1}^n \alpha_i e^{\lambda_i t} v_i$$

where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)^T = M^{-1}x_0$.

Thus, from above, we see that the **eigenvalues determine the rate at which the system response decays or grows** and the **eigenvectors determine the shape of the response**. The problem of assigning both eigenvalues and eigenvectors is called the **eigenstructure assignment problem**.

Formally, the problem is stated as follows:

Given the sets $S = \{\mu_1, \dots, \mu_n\}$ and $M = \{v_1, \dots, v_n\}$ of scalars and vectors, respectively, both closed under complex conjugation, find a feedback matrix K such that the matrix $A + BK$ has the μ_i 's as the eigenvalues and the v_i 's as the corresponding eigenvectors.

The following result, due to Moore (1976), gives a necessary and sufficient condition for a solution of the eigenstructure assignment problem by state feedback (see Andry, Shapiro and Chung (1983) for details and a proof).

Define $R_\lambda = \begin{bmatrix} N_\lambda \\ M_\lambda \end{bmatrix}$, where the columns of R_λ form a basis for the null space of the matrix $(\lambda I - A, B)$.

Theorem 10.4.3 (The State Feedback Eigenstructure Assignment Theorem)

Assume that the numbers $\{\mu_i\}$ in the set S are distinct and self-conjugate. Then there exists a matrix K such that $(A + BK)v_i = \mu_i v_i, i = 1, \dots, n$ if and only if, the following conditions are satisfied:

- (i) The vectors v_1, \dots, v_n are linearly independent
- (ii) $v_i = v_j^*$ whenever $\mu_i = \mu_j^*, i = 1, 2, \dots, n$
- (iii) $v_i \in \text{span}\{N_{\mu_i}\}, i = 1, 2, \dots, n$

If B has full rank and K exists, then it is unique. When μ_i 's are all real and distinct, an expression for K is

$$K = (-M_{\mu_1}z_1, -M_{\mu_2}z_2, \dots, -M_{\mu_n}z_n)(v_1, v_2, \dots, v_n)^{-1},$$

where the vector z_i is given by

$$v_i = N_{\mu_i}z_i, \quad i = 1, 2, \dots, n.$$

■

The following result on the eigenstructure assignment by output feedback is due to Srinathkumar (1978).

Theorem 10.4.4 (The Output Feedback Eigenstructure Assignment Theorem)

Let (A, B) be controllable and (A, C) be observable. Assume that $\text{rank}(B) = m$ and $\text{rank}(C) = r$. Then $\max(m, r)$ eigenvalues and $\max(m, r)$ eigenvectors with $\min(m, r)$ entries in each eigenvector can be assigned by the output feedback law (10.4.8).

Note: Numerically effective algorithms for the output feedback problem are rare. Perhaps, the first comprehensive work in this context is the paper by Misra and Patel (1989), where algorithms for both the single-input and the multi-output systems, using implicit shifts, have been given. We refer the readers to the above paper for a description of this algorithm.

10.5 The Quadratic Optimization Problems

We have just seen that if a system is controllable, then the closed-loop eigenvalues can be placed at arbitrarily chosen locations of the complex plane. But, the lack of the existence of a definite guideline of where to place these eigenvalues makes the design procedure a rather difficult one in practice. A designer has to use his or her own intuition of how to use the freedom of choosing the eigenvalues to achieve the design objective.

It is, therefore, desirable to have a design method that can be used as an initial design process while the designer develops his or her insight.

A “compromise” is often made in practice to obtain such an initial design process. Instead of trying to place the eigenvalues at desired locations, the system is stabilized while satisfying certain performance criterion.

Specifically, the following problem, known as the **Linear Quadratic Optimization Problem**, is solved. The problem is also commonly known as the “**Linear Quadratic Regulator**” (LQR) problem.

10.5.1 The Continuous-time Linear Quadratic Regulator (LQR) Problem

Given matrices Q and R , find a control signal $u(t)$ such that the quadratic cost function $J_C(x) = \int_0^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)] dt$ is minimized, subject to $\dot{x} = Ax + Bu, x(0) = x_0$.

The matrices Q and R represent respectively weights for the states and the control vectors. The quadratic form $x^T Q x$ represents the deviation of the state x from the initial state, and the term $u^T R u$ represents the “cost” of control. The matrices Q and R need to be chosen according to the requirements of a specific design. Note that the magnitude of the control signal u can be properly controlled by choosing R appropriately. In fact, by selecting large R , $u(t)$ can be made small (see the expression of the unique control law in Theorem 10.5.1), which is desirable. The choice of Q is related to which states are to be kept small.

Unfortunately, again it is hard to set a specific guideline of how to choose Q and R . “The choice of these quantities is again more of an art than a science” (Kailath (1980), pp. 219). For a meaningful optimization problem, however, it is assumed that **Q is symmetric positive semidefinite and R symmetric positive definite**. Unless mentioned otherwise, we will make these assumptions throughout the rest of the chapter.

The solution of the above problem can be obtained via the solution of a quadratic matrix equation called the **algebraic Riccati equation** (ARE), as shown by the following result. See Anderson and Moore (1990) for details.

Theorem 10.5.1 (The Continuous-Time LQR Theorem) *Suppose the pair (A, B) is stabilizable and the pair (A, Q) is detectable. Then there exists a unique optimal control $u^0(t)$ which minimizes $J_C(x)$. The vector $u^0(t)$ is given by $u^0(t) = -Kx(t)$, where $K = R^{-1}B^TX$, and X is the unique positive semidefinite solution of the algebraic Riccati equation*

$$XA + A^T X + Q - XBR^{-1}B^T X = 0. \quad (10.5.1)$$

Furthermore, the closed-loop matrix $A - BK$ is stable and the minimum value of $J_C(x)$ is equal to $x_0^T X x_0$, where $x_0 = x(0)$.

■

The proof of the existence and uniqueness of the stabilizing solution (under the conditions that (A, B) is stabilizable and (A, Q) is detectable) will be deferred until Chapter 13. Here we give a proof of the optimal control part, assuming that such a solution exists.

Proof: Proof of the Optimal Control Part of Theorem 10.5.1

$$\begin{aligned}
\frac{d}{dt}(x^T X x) &= \dot{x}^T X x + x^T X \dot{x} \\
&= (Ax + Bu)^T X x + x^T X (Ax + Bu) \\
&= (u^T B^T + x^T A^T) X x + x^T X (Ax + Bu) \\
&= x^T (A^T X + X A) x + u^T B^T X x + x^T X B u \\
&= x^T (X B R^{-1} B^T X - Q) x + u^T B^T X x + x^T X B u \quad (\text{using (10.5.1)}) \\
&= x^T X B R^{-1} B^T X x + u^T B^T X x + x^T X B u + u^T R u \\
&\quad - u^T R u - x^T Q x \\
&= (u^T + x^T X B R^{-1}) R (u + R^{-1} B^T X x) - (x^T Q x + u^T R u)
\end{aligned}$$

or

$$x^T Q x + u^T R u = -\frac{d}{dt}(x^T X x) + (u^T + x^T X B R^{-1}) R (u + R^{-1} B^T X x)$$

Integrating with respect to t from 0 to T , we obtain

$$\int_0^T (x^T Q x + u^T R u) dt = -x^T(T) X x(T) + x_0^T X x_0 + \int_0^T (u + R^{-1} B^T X x)^T R (u + R^{-1} B^T X x) dt$$

(Note that $X = X^T \geq 0$ and $R = R^T > 0$).

Letting $T \rightarrow \infty$ and noting that $x(T) \rightarrow 0$ as $T \rightarrow \infty$, we obtain

$$J_C(x) = x_0^T X x_0 + \int_0^\infty (u + R^{-1} B^T X x)^T R (u + R^{-1} B^T X x) dt$$

Since R is symmetric and positive definite, it follows that $J_C(x) \geq x_0^T X x_0$ for all x_0 and for all controls u . Since the first term $x_0^T X x_0$ is independent of u , the minimum value of $J_C(x)$ occurs at

$$u^0(t) = -R^{-1} B^T X x(t) = -K x(t).$$

The minimum value of $J_C(x)$ is therefore $x_0^T X x_0$.

■

Definition 10.5.1 *The algebraic Riccati equation*

$$XA + A^T X + Q - X S X = 0, \quad (10.5.2)$$

where $S = BR^{-1}B^T$ is called the **Continuous-Time Algebraic Riccati Equation** or in short **CARE**.

Definition 10.5.2 *The matrix H defined by*

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} \quad (10.5.3)$$

is the Hamiltonian matrix associated with the CARE (10.5.2).

Definition 10.5.3 A symmetric solution X of the CARE such that $A - SX$ is stable is called a stabilizing solution.

Relationship Between Hamiltonian Matrix and Riccati Equations

The following theorem shows that there exists a very important relationship between the Hamiltonian matrix (10.5.3) and the CARE (10.5.2). The proof will be deferred until Chapter 13.

Theorem 10.5.2 Let (A, B) be stabilizable and (A, Q) be detectable. Then the Hamiltonian matrix H in (10.5.3) has n eigenvalues with negative real parts, no eigenvalues on the imaginary axis and n eigenvalues with positive real parts. In this case the CARE (10.5.2) has a unique stabilizing solution X . Furthermore, the closed-loop eigenvalues, that is, the eigenvalues of $A - BK$, are the stable eigenvalues of H .

A Note on the Solution of the CARE:

It will be shown in **Chapter 13** that the unique stabilizing solution to (10.5.2) can be obtained by constructing an invariant subspace associated with the stable eigenvalues of the Hamiltonian matrix H in (10.5.3). Specifically, if H does not have any imaginary eigenvalue and $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ is the matrix with columns composed of the eigenvectors corresponding to the stable eigenvalues of H , then, assuming that X_1 is nonsingular, the matrix $X = X_2X_1^{-1}$ is a unique stabilizing solution of the CARE. For details, see **Chapter 13**.

The MATLAB function **care** solves the CARE. The matrix S in CARE is assumed to be nonnegative definite.

The Continuous-time LQR Design Algorithm

From Theorem 10.5.1, we immediately have the following LQR design algorithm.

Algorithm 10.5.1 The Continuous-time LQR Design Algorithm

Inputs: The matrices A, B, Q, R , and $x(0) = x_0$.

Outputs: X —The solution of the CARE

K —The LQR feedback gain matrix

$J_{c\min}$ —The minimum value of the cost function $J_C(x)$.

Assumptions:

1. (A, B) is stabilizable and (A, Q) is detectable.
2. Q is symmetric positive semidefinite and R is symmetric positive definite.

Step 1. Compute the stabilizing solution X of the CARE:

$$XA + A^T X - XSX + Q = 0, \quad S = BR^{-1}B^T.$$

Step 2. Compute the LQR feedback gain matrix:

$$K = R^{-1}B^T X$$

Step 3. Compute the minimum value of $J_C(x) : J_{c\min} = x_0^T X x_0$.

Example 10.5.1 (LQR Design for the Inverted Pendulum). We consider Example 10.2.1 again, with A and B , the same as there and $Q = I_4$, $R = 1$, and $x_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

Step 1. The unique positive definite solution X of the CARE (obtained by using MATLAB function `care`) is

$$X = 10^3 \begin{pmatrix} 0.0031 & 0.0042 & 0.0288 & 0.0067 \\ 0.0042 & 0.0115 & 0.0818 & 0.0191 \\ 0.0288 & 0.0818 & 1.8856 & 0.4138 \\ 0.0067 & 0.0191 & 0.4138 & 0.0911 \end{pmatrix}.$$

Step 2. The feedback gain matrix K is

$$K = (-1, -3.0766, -132.7953, -28.7861).$$

Step 3. The minimum value of $J_C(x)$ is 3100.3.

The eigenvalues of $A - BK$ are: $-4.8994, -4.5020, -0.4412 \pm 0.3718j$. Thus X is the unique positive definite stabilizing solution of the CARE.

(Note that the entries of K in this case are smaller compared to those of K in Example 10.2.1).

Comparison of Transient Responses with Lyapunov Stabilization

The graphs below show the transient responses of the closed-loop systems with (i) K from Example 10.2.1 and (ii) K as obtained above. The initial condition $x(0) = (5, 0, 0, 0)^T$.

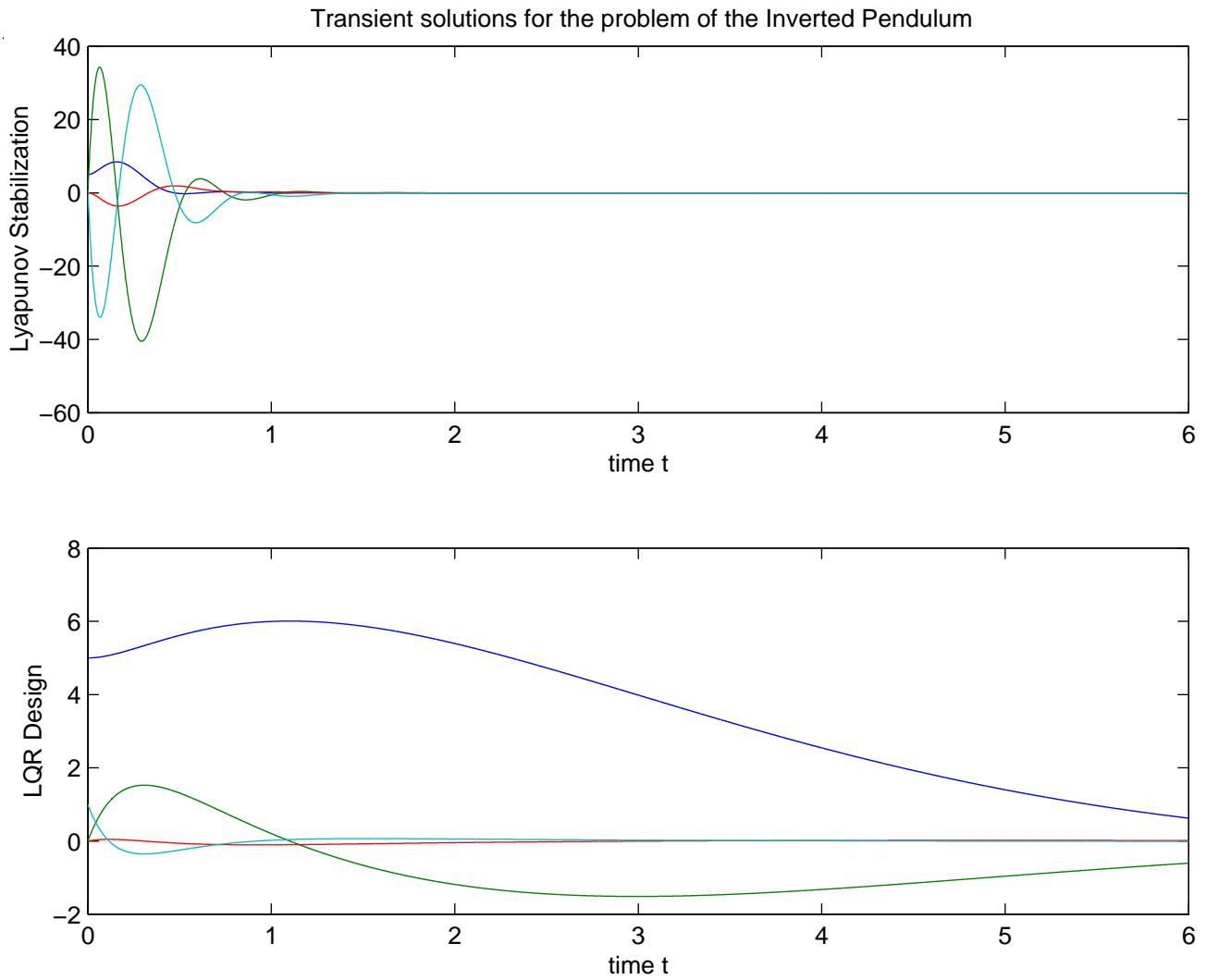


Figure 10.7 Transient Responses (The Upper Graph Corresponds to Lyapunov Method and the Lower Graph Corresponds to LQR Design).

In the first case, the transient solutions initially have large magnitudes and then they decay rapidly. In the second case, the solutions have smaller magnitudes but the decay rate is much slower. The largest magnitude in transient solution in the first case is roughly six times larger than the one in the second case. In some dynamical systems, strong initial oscillations in the state components must be avoided, but sometimes a faster stabilization is desired; in other cases, a slow but smooth stabilization is required.

Note that the transient solutions in the first case, however, depend upon β and in the second case depend upon Q and R (see below).

Stability and Robustness Properties of the LQR Design

The LQR design has some very favourable stability and robustness properties. We will list some important ones here.

Guaranteed Stability Properties

Clearly, the closed-loop eigenvalues of the LQR design depend upon the matrices Q and R . We will show here how the choice of R affects the closed-loop poles.

Suppose $R = \rho I$, where ρ is a positive scalar. Then, the associated Hamiltonian matrix

$$H = \begin{pmatrix} A & -\frac{1}{\rho}BB^T \\ -Q & -A^T \end{pmatrix}.$$

The closed-loop eigenvalues are the roots with negative real parts of the characteristic polynomial

$$d_c(s) = \det(sI - H).$$

Let $Q = C^T C$. It can be shown that

$$d_c(s) = (-1)^n d(s) d(-s) \det[I + \frac{1}{\rho} G(s) G^T(-s)],$$

where $d(s) = \det(sI - A)$, and $G(s) = C(sI - A)^{-1}B$.

Case 1. Low Gain

When $\rho \rightarrow \infty$, $u(t) = -\frac{1}{\rho} B^T X x(t) \rightarrow 0$. Thus, the LQR controller has low gain. In this case, from the above expression of $d_c(s)$, it follows that

$$(-1)^n d_c(s) \rightarrow d(s) d(-s).$$

Since the roots of $d_c(s)$, that is, the closed-loop eigenvalues, are stable, this means that **as ρ increases**

- The stable open-loop eigenvalues remain stable.
- The unstable ones get reflected across the imaginary axis.
- If any open-loop eigenvalues are exactly on the $j\omega$ -axis, the closed-loop eigenvalues start moving just left of them.

Case 2. High Gain.

If $\rho \rightarrow 0$, then $u(t)$ becomes large; thus, the LQR controller has high gain.

In this case, for finite s , the closed-loop eigenvalues approach the finite zeros of the system or their stable images.

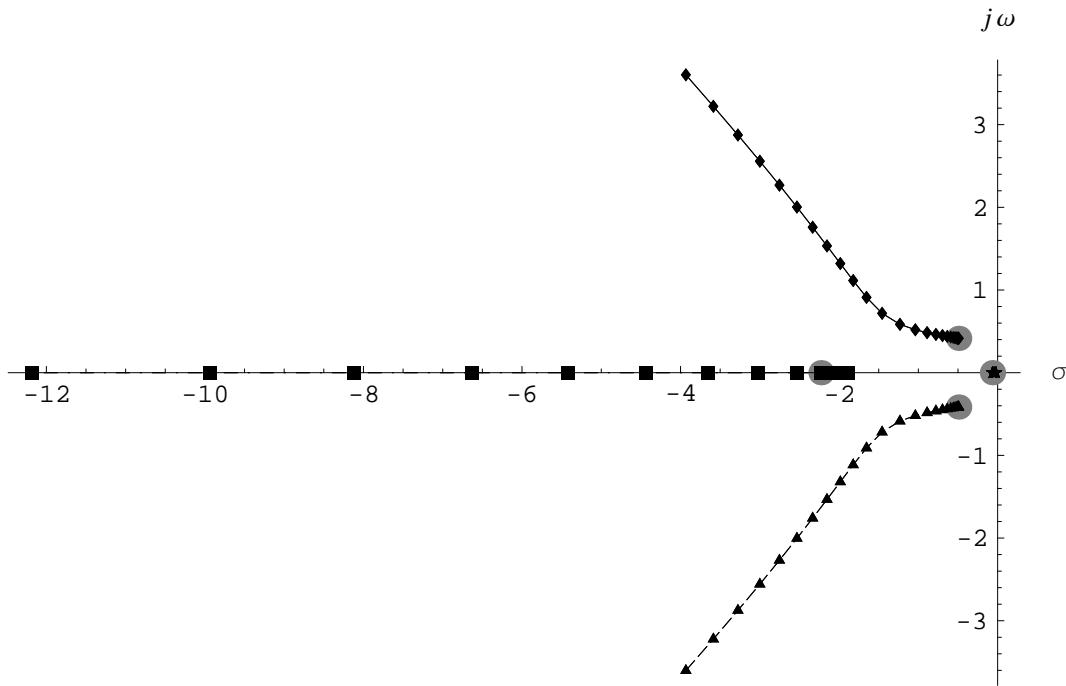


Figure 10.4: Illustration of Butterworth Patterns.

As $s \rightarrow \infty$, the closed-loop eigenvalues will approach infinity in the so-called stable **Butterworth patterns**. (For a description of Butterworth patterns, see Friedland (1986)). An example is given in the above diagram.

These properties, provide good insight into the stability property of LQR controllers and, thus, can be used by a designer as a guideline of where to place the poles.

Robustness Properties of the LQR Design

As we have seen before, an important requirement of a control system design is that the closed-loop system be **robust** to uncertainties due to modeling errors, noise and disturbances. It is shown below that the LQR design has some desirable robustness properties.

The classical robust stability measures are **gain** and **phase margins**, defined with respect to the **Bode plot** (see **Chapter 5**) of a SISO system.

The **gain margin** is defined to be the amount of gain increase required to make the loop gain unity where the phase angle is 180° . That is, **it is the reciprocal of the gain at the frequency where the phase angle is 180°** . Thus, the *gain margin is a measure by which the system gain needs to be increased for the closed-loop system to become just unstable*.

Similarly, the difference between the phase of the response and -180° when the loop gain is unity is called the **phase margin**. *That is, the phase margin is the minimum amount of phase lag that is needed to make the system unstable.*

The robustness properties of the LQR design for MIMO system can be studied by considering the **return difference matrix**: $I + G_{LQ}(s)$, where $G_{LQ}(s)$ is the LQR loop transfer function matrix given by

$$G_{LQ}(s) = K(sI - A)^{-1}B.$$

The **optimal return difference identity** is:

$$[I + K(-sI - A)^{-1}B]^T R [I + K(sI - A)^{-1}B] = R + B^T (-sI - A)^{-T} Q (sI - A)^{-1} B.$$

or

$$(I + G_{LQ}^T(-s))R(I + G_{LQ}(s)) = R + G^T(-s)QG(s).$$

From the above equation, we have

$$(I + G_{LQ}^*(j\omega))R(I + G_{LQ}(j\omega)) \geq R.$$

It has been shown in Safonov and Athans (1977) that if R is a diagonal matrix so that

$$(I + G_{LQ}^*(j\omega))(I + G_{LQ}(j\omega)) \geq I$$

then there is at least 60° of phase margin in each input channel and the gain margin is infinity. *This means that a phase shift of up to 60° can be tolerated in each of the input channels simultaneously and the gain in each channel can be increased indefinitely without losing stability.* It also follows from above [**Exercise 20 (b)**] that

$$\sigma_{\min}(I + G_{LQ}(j\omega)) \geq 1.$$

This means that the LQ design always results in decreased sensitivity.

See Anderson and Moore (1990, 122-135), the article “Optimal Control” by F. L. Lewis in the **Control Handbook** (1996, 759-778) edited by William Levine, IEEE/CRC Press, and Lewis (1986, 1992), and Maciejowski (1989) for further details.

Example 10.5.2 Consider Example 10.5.1 again, For $\omega = 1$, $G_{LQ}(j\omega) = -1.9700 + 0.5345j$

$$\sigma_{\min}(1 + G_{LQ}(j\omega)) = 1.1076$$

$$\sigma_{\min}(1 + G_{LQ}^{-1}(j\omega)) = 0.5426$$

$$\text{The gain margin} = 0.4907$$

$$\text{The phase margin} = 60.0998.$$

LQR Stability with Multiplicative Uncertainty

The inequality

$$\sigma_{\min}(I + G_{LQ}(j\omega)) \geq 1$$

also implies [Exercise 20(c)] that

$$\sigma_{\min}(I + (G_{LQ}(j\omega))^{-1}) \geq \frac{1}{2}$$

which means that LQR design remains stable for all unmeasured multiplicative uncertainties Δ in the system for which $\sigma_{\min}(\Delta(j\omega)) \leq \frac{1}{2}$.

MATLAB Notes: The MATLAB command $[K, S, E] = lqr(A, B, Q, R)$ solves the LQR problem. Here

K —Feedback matrix

S —Steady-state solution of the CARE

$$SA + A^T S - SBR^{-1}B^T S + Q = 0.$$

E —The vector containing the closed-loop eigenvalues.

The CARE is solved by using the generalized **Schur algorithm** to be described in Chapter 13.

MATLAB function **margin** can be used to compute the gain and phase margins of a system.

10.5.2 The Discrete-time Linear Quadratic Regulator Problem

In the discrete-time case, the function to be minimized is:

$$J_D(x) = \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k), Q \geq 0, R \geq 0. \quad (10.5.4)$$

and the associated algebraic Riccati equation is:

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0 \quad (10.5.5)$$

The above equation is called the **Discrete-time Algebraic Riccati Equation** (DARE).

A theorem on the existence and uniqueness of the optimal control u_k^0 , similar to Theorem 10.5.1, is stated next. For a proof, see Sage and White (1977).

Theorem 10.5.3 (The Discrete-Time LQR Theorem) *Let (A, B) be discrete-stabilizable and (A, Q) be discrete-detectable. Then the optimal control $u_k^0, k = 0, 1, 2, \dots$, that minimizes $J_D(x)$ is given by $u_k^0 = -K x_k$, where $K = (R + B^T X B)^{-1} B^T X A$, and X is the unique positive semidefinite solution of the DARE (10.5.5). Furthermore, the closed-loop discrete system*

$$x_{k+1} = (A - BK) x_k$$

is discrete-stable (that is, all the eigenvalues are strictly inside the unit circle), and the minimum value of $J_D(x)$ is $x_0^T X x_0$, where x_0 is the given initial state.

Definition 10.5.4 A symmetric solution X of the DARE that makes the matrix $A - BK$, where $K = (R + B^T X B)^{-1} B^T X A$, discrete-stable is called a **discrete-stabilizing solution** of the DARE.

Example 10.5.3 $A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}$, $B = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, $Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $R = 1$.

The solution X of the DARE (computed by using MATLAB function **dlqr**) is:

$$X = 10^3 \begin{pmatrix} 0.0051 & -0.0542 & 0.0421 \\ -0.542 & 1.0954 & -0.9344 \\ 0.0421 & -0.9344 & 0.8127 \end{pmatrix}.$$

The discrete LQR gain matrix

$$K = (-0.0437, \quad 2.5872, \quad -3.4543).$$

The eigenvalues of $A - BK$ are: $-0.4266, -0.2186, -0.1228$. Thus, X is a discrete-stabilizing solution.

MATLAB Note: The MATLAB function **lqr** computes the discrete-time feedback-gain matrix given in Theorem 10.5.3.

10.6 H_∞ Control Problems

So far we have considered the stabilization of a system ignoring any effect of disturbances in the system. But, we already know that in practice a system is always acted upon by some kind of disturbances. Thus, it is desirable to stabilize perturbed versions of a system, assuming certain bounds for perturbations. This situation gives rise to the well-known “ **H_∞ control problem**.”

H_∞ control theory has been the subject of intensive study for the last twenty years or so, since its introduction by Zames (1981). There are now excellent literature in the area: the books by Francis (1987), Kimura (1996), Zhou, Doyle and Glover (1996), Green and Limebeer (1995), etc., and the original important papers by Francis and Doyle (1987), Doyle, Glover, Khargonekar and Francis (1989), etc.

Let $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$ denote, respectively, the largest and smallest singular value of M .

Definition 10.6.1 The H_∞ -norm of the stable transfer function $G(s)$, denoted by $\|G\|_\infty$, is defined by

$$\|G\|_\infty = \sup_{\omega \in \mathbb{R}} \sigma_{\max}(G(j\omega)).$$

In the above definition, “sup” means the supremum or the least upper bound of the function $\sigma_{\max}(G(j\omega))$.

Physical Interpretation of the H_∞ -norm.

Consider the system

$$y(s) = G(s)u(s).$$

Then when the system is driven with a sinusoidal input of unit magnitude at a specific frequency, $\sigma_{\max}(G(j\omega))$ is the largest possible size of the output for the corresponding sinusoidal input. Thus, the H_∞ -norm gives the largest possible amplification over all frequencies of a unit sinusoidal input.

A detailed discussion of H_∞ control problems is beyond the scope of this book. The original formulation was in an input/output setting. However, due to its computational attractiveness, the recent state-space formulation has become more popular. We only state two **simplified versions** of the state-space formulations of H_∞ control problems, and mention their connections with algebraic Riccati equations. First, we prove the following well-known result that shows how the H_∞ -norm of a stable transfer function matrix is related to an algebraic Riccati equation or equivalently, to the associated Hamiltonian matrix.

Define the **Hamiltonian matrix** out of the coefficients of the matrices of the system (10.2.1)

$$M_\gamma = \begin{pmatrix} A + BR^{-1}D^T C & BR^{-1}B^T \\ -C^T(I + DR^{-1}D^T)C & -(A + BR^{-1}D^T C)^T \end{pmatrix} \quad (10.6.1)$$

where $R = \gamma^2 I - D^T D$.

Theorem 10.6.1 Let $G(s)$ be a stable transfer function and let $\gamma > 0$. Then $\|G\|_\infty < \gamma$ if and only if $\sigma_{\max}(D) < \gamma$ and M_γ defined by (10.6.1) has no imaginary eigenvalues.

Proof: We sketch the proof in case $D = 0$. This proof can easily be extended to the case when $D \neq 0$, and is left as an exercise [Exercise 23]. Without any loss of generality, assume that $\gamma = 1$. Otherwise, we can scale G to $\gamma^{-1}G$ and B to $\gamma^{-1}B$ and work with scaled G and B (note that $\|G\|_\infty < \gamma$ if and only if $\|\gamma^{-1}G\|_\infty < 1$).

Since $\gamma = 1$ and $D = 0$, we have $R = I$. Using the notation

$$G(s) = C(sI - A)^{-1}B \equiv \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array} \right],$$

an easy computation shows that if

$$\Gamma(s) = I - G(-s)^T G(s), \quad (10.6.2)$$

then

$$\Gamma^{-1}(s) = \left[\begin{array}{cc|c} A & BB^T & B \\ -C^T C & -A^T & 0 \\ \hline 0 & B^T & I \end{array} \right] = \left[\begin{array}{c|c} M_\gamma & B \\ \hline 0 & 0 \end{array} \right]. \quad (10.6.3)$$

From above, it, therefore, follows that M_γ does not have an eigenvalue on the imaginary axis if and only if $\Gamma^{-1}(s)$ does not have any pole there. We now show that this is true if and only if $\|G\|_\infty$ is less than 1.

If $\|G\|_\infty < 1$, then $I - G(j\omega)^*G(j\omega) > 0$ for every ω , and hence $\Gamma^{-1}(s) = (I - G(-s)^T G(s))^{-1}$ does not have any pole on the imaginary axis. On the other hand, if $\|G\|_\infty \geq 1$, then $\sigma_{max}(G(j\omega)) = 1$ for some ω , which means that 1 is an eigenvalue of $G(j\omega)^*G(j\omega)$, implying that $I - G(j\omega)^*G(j\omega)$ is singular. ■

The following simple example from Kimura (1996, pp. 41) illustrates Theorem 10.6.1.

Example 10.6.1 Let

$$G(s) = \frac{1}{s + \alpha}, \quad \alpha > 0.$$

The associated Hamiltonian matrix

$$H = \begin{pmatrix} -\alpha & 1 \\ -1 & \alpha \end{pmatrix}.$$

Then H does not have any imaginary eigenvalue if and only if $\alpha > 1$.

Since $\|G\|_\infty = \sup_\omega \frac{1}{\sqrt{\omega^2 + \alpha^2}} = \frac{1}{\alpha}$, we have, for $\alpha > 1$, $\|G\|_\infty < 1$.

10.6.1 Computing the H_∞ -Norm

A straightforward way to compute the H_∞ -norm is:

1. Compute the matrix $G(j\omega)$ using the Hessenberg method described in Chapter 5.
2. Compute the largest singular value of $G(j\omega)$
3. Repeat the steps 1 and 2 for many values of ω .

Certainly the above approach is impractical and computationally infeasible.

However, Theorem 10.6.1 gives us a more practically feasible method for computing the H_∞ -norm. The method then will be as follows:

1. Choose γ .
2. Test if $\|G\|_\infty < \gamma$, by computing the eigenvalues of M_γ and seeing if the matrix M_γ has an imaginary eigenvalue.
3. Repeat, by increasing or decreasing γ , according as $\|G\|_\infty < \gamma$ or $\|G\|_\infty \geq \gamma$.

The following bisection method of Boyd et al. (1989) is an efficient and systematic implementation of the above idea.

Algorithm 10.6.1 The Bisection Algorithm for Computing the H_∞ -Norm

Inputs: The system matrices A, B, C and D , of dimensions $n \times n, n \times m, r \times n$, and $r \times m$, respectively.

γ_{lb} - A lower bound of the H_∞ -norm

γ_{ub} - An upper bound of the H_∞ -norm

$\epsilon (> 0)$ - Error tolerance

Output: An approximation of the H_∞ -norm with a relative accuracy of ϵ .

Assumption: A is stable.

Step 1. Set $\gamma_L \equiv \gamma_{lb}$, and $\gamma_U = \gamma_{ub}$

Step 2. Repeat until $\gamma_U - \gamma_L \leq 2\epsilon\gamma_L$

Compute $\gamma = (\gamma_L + \gamma_U)/2$

Test if M_γ defined by (10.6.1) has an imaginary eigenvalue

If not, set $\gamma_U = \gamma$

Else, set $\gamma_L = \gamma$

Remark: After k iterations, we have $\gamma_U - \gamma_L = 2^{-k}(\gamma_{ub} - \gamma_{lb})$. Thus on exit, the Algorithm is guaranteed to give an approximation of the H_∞ -norm with a relative accuracy of ϵ .

Convergence. The convergence of the algorithm is **linear** and is independent of the data matrices A, B, C , and D .

Note: An algorithm equivalent to the above bisection algorithm was also proposed by Robel (1989).

Remark: The condition that M_γ does not have an imaginary eigenvalue (in Step 2) can also be expressed in terms of the associated Riccati equation:

$$XA + A^T X + \gamma^{-1} XBR^{-1}B^T X + \gamma^{-1}C^T C = 0$$

(Assuming that $D = 0$).

Example 10.6.2

$$A = \begin{pmatrix} -1 & 2 & 3 \\ 0 & -2 & 0 \\ 0 & 0 & -4 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad C = (1, \ 1, \ 1), \quad D = 0, \quad \epsilon = 0.0014$$

Step 1: $\gamma_L = 0.2887, \ \gamma_U = 1.7321$.

Step 2:

Iteration 1

$$\gamma = 1.0104.$$

The eigenvalues of M_γ are $\{2, -4, -0.1429, 0.1429, -2.0000, -4, 0000\}$. Since M_γ does not have purely imaginary eigenvalues, we continue.

$$\gamma_L = 0.28867, \quad \gamma_U = 1.0103$$

Iteration 2

$$\gamma = 0.6495$$

The eigenvalues of M_γ are $\{2, 4, -2, -4, -0 \pm 1.1706j\}$. Since M_γ has a purely imaginary eigenvalue, we set

$$\gamma_L = 0.6495, \quad \gamma_U = 1.0103,$$

Iteration 3

$$\gamma = 0.8299$$

The eigenvalues of M_γ are $\{2, 4, -2, -4, 0 \pm 0.6722j\}$. Since M_γ has a purely imaginary eigenvalue, we set

$$\gamma_L = 0.8299, \quad \gamma_U = 1.0103.$$

Continuing this way, after 11 iterations, we obtain $\gamma_L = 0.9990$ and $\gamma_U = 1.0004$. Since $\gamma_L - \gamma_U = 0.0014 < 2\epsilon\gamma_L = 0.0028$, we stop n and take $\gamma = H_\infty - \text{norm} = 1.0005$.

Computing γ_{lb} and γ_{ub}

For practical implementation of the above algorithm, we need to know how to compute γ_{lb} and γ_{ub} . We will discuss this aspect now.

Definition 10.6.2 The **Hankel singular values** are the square roots of the eigenvalues of the matrix $C_G O_G$, where C_G and O_G are the controllability and observability Grammians, respectively.

The bounds γ_{lb} and γ_{ub} may be computed using the *Hankel singular values* as follows:

$$\begin{aligned} \gamma_{lb} &= \max\{\sigma_{\max}(D), \sigma H_1\}, \\ \gamma_{ub} &= \sigma_{\max}(D) + 2 \sum_{i=1}^n \sigma H_i \end{aligned} \tag{10.6.4}$$

where σH_k 's are the ordered **Hankel singular values**, that is, σH_i is the i -th largest Hankel singular value. These bounds are due to Enns (1984) and Glover (1984) and the formula (10.6.4) is known as the **Enns-Glover formula**.

A scheme for computing γ_{lb} and γ_{ub} will then be as follows:

1. Solve the Lyapunov equations (7.2.11) and (7.2.12) to obtain C_G and O_G , respectively.

2. Compute the eigenvalues of $C_G O_G$.
3. Obtain the Hankel singular values by taking the square roots of the eigenvalues of $C_G O_G$.
4. Compute γ_{lb} and γ_{ub} using the above Enns-Glover formula.

As an alternative to eigenvalue computations, one can also use the following formulas:

$$\begin{aligned}\gamma_{lb} &= \max\{\sigma_{\max}(D), \sqrt{\text{Trace}(C_G O_G)/n}\} \\ \gamma_{ub} &= \sigma_{\max}(D) + 2\sqrt{n\text{Trace}(C_G O_G)}.\end{aligned}$$

Remarks: Numerical difficulties can be expected in forming the product $C_G O_G$ explicitly.

MATCONTROL NOTE: Algorithm 10.6.1 has been implemented in MATCONTROL function **hinfmr**.

The Two-Step Algorithm

Recently, Bruinsma and Steinbuch (1990) have developed a “**two-step**” algorithm to compute H_∞ -norm of $G(s)$. Their algorithm is faster than the bisection algorithm just stated. **The convergence is claimed to be quadratic.**

The algorithm is called a “two-step” algorithm, because, the algorithm starts with some lower bound of $\gamma < \|G\|_\infty$, as the first step and then in the second step, this lower bound is iteratively improved and the procedure is continued until some “tolerance” is satisfied.

A New Lower Bound of the H_∞ -norm.

The two-step algorithm, like the bisection algorithm, requires a starting value for γ_{lb} . The Enns-Glover formula can be used for this purpose. However, the authors have proposed that the following starting value for γ_{lb} be used:

$$\gamma_{lb} = \max\{\sigma_{\max}(G(0)), \sigma_{\max}(G(j\omega_p)), \sigma_{\max}(D)\},$$

where $\omega_p = |\lambda_i|$, λ_i a pole of $G(s)$ with λ_i selected to maximize $\left| \frac{\text{Im}(\lambda_i)}{\text{Re}(\lambda_i)} \frac{1}{|\lambda_i|} \right|$, if $G(s)$ has poles with $\text{Im}(\lambda_i) \neq 0$ or to minimize $|\lambda_i|$, if $G(s)$ has only real poles.

Algorithm 10.6.2 The Two-Step Algorithm For Computing the H_∞ -norm.

Inputs: The system matrices A, B, C and D , respectively, of dimensions $n \times n, n \times m, r \times n$, and $r \times m$. ϵ - Error tolerance.

Output: An approximate value of the H_∞ -norm.

Assumption. A is stable.

Step 1. Compute a starting value for γ_{lb} .

Step 2. Repeat

2.1 Compute $\gamma = (1 + 2\epsilon)\gamma_{lb}$.

2.2 Compute the eigenvalues of M_γ , with the value of γ computed in Step

2.1. Label the purely imaginary eigenvalues of M_γ as $\omega_1, \dots, \omega_k$.

2.3 If M_γ does not have a purely imaginary eigenvalue, set $\gamma_{ub} = \gamma$ and stop.

2.4 For $i = 1, \dots, k - 1$ do

(a) Compute $m_i = \frac{1}{2}(\omega_i + \omega_{i+1})$.

(b) Compute the singular values of $G(jm_i)$.

End

2.5 Update $\gamma_{lb} = \max_i(\sigma_{\max}(G(jm_i)))$.

Step 3. $\| G \|_\infty = \frac{1}{2}(\gamma_{lb} + \gamma_{ub})$.

MATLAB NOTE: Algorithm 10.6.2 has been implemented in MATLAB Control System tool box. The usage is: **norm** (sys,inf).

In the above, “sys” stands for the linear time-invariant system in the matrices A, B, C and D . “sys” can be generated as follows:

$$A = [], \quad B = [], \quad C = [], \quad D = [], \quad \text{sys} = ss(A, B, C, D).$$

Remarks: Boyd and Balakrishnan (1990) have also proposed an algorithm similar to the above “two-step” algorithm. Their algorithm converges quadratically. Algorithm 10.6.2 also is believed to converge quadratically, but no proof has been given. See also the paper of Lin, Wang, and Xu (1999) for other recent reliable and efficient methods for computing the H_∞ norms for both the state and output feedback problems.

Connection between H_∞ -norm and the Distance to Unstable Matrices

Here we point out a connection between the H_∞ -norm of the resolvent of A and $\beta(A)$, the distance to the set of unstable matrices. The proof is easy and left as an [**Exercise 27**].

Theorem 10.6.2 Let A be a complex stable matrix, then $\beta(A) = \| (sI - A)^{-1} \|_\infty^{-1}$. ■

Computing the H_∞ -Norm of a Discrete-Stable Transfer Function Matrix

Let $M(z) = C(zI - A)^{-1}B$ be the transfer function matrix of the **asymptotically stable discrete-time system**:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k. \end{aligned}$$

Then

Definition 10.6.3 The H_∞ -norm of $M(z)$ is defined as

$$\| M(z) \|_\infty = \sup_{|z| \geq 1} \sigma_{\max}(M(z)).$$

The following is a discrete-analog of Theorem 10.6.1. We state the result here without proof. For proof, we refer the readers to the book by Zhou, Doyle and Glover (1996, pp. 547-548).

Theorem 10.6.3 Let

$$S = \begin{pmatrix} A - BB^T(A^T)^{-1}C^TC & BB^T(A^T)^{-1} \\ -(A^T)^{-1}C^TC & (A^T)^{-1} \end{pmatrix}$$

be the symplectic matrix associated with the above stable discrete-time system. Assume that A is nonsingular and that the system does not have any uncontrollable and unobservable modes on the unit circle.

Then $\| M(z) \|_\infty < 1$ if and only if S has no eigenvalues on the unit circle and $\| C(I - A)^{-1}B \|_2 < 1$.

■

Computing H_∞ -norm of a discrete-stable system.

The above theorem can now be used as a basis to develop a **bisection algorithm**, analogous to Algorithm 10.6.1, for computing the H_∞ -norm of a discrete stable system. We leave this an exercise [**Exercise 24**].

10.6.2 H_∞ Control Problem: A State Feedback Case.

Consider the following linear control system:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + Ed(t), \quad x(0) = 0 \\ z(t) &= Cx(t) + Du(t). \end{aligned} \tag{10.6.5}$$

Here $x(t)$, $u(t)$, and $z(t)$, denote the state, the control input, and controlled output vectors. The vector $d(t)$ is the disturbance vector. The matrices A, B, C, D and E are matrices of appropriate dimensions. Suppose that a state feedback control law

$$u(t) = Kx(t)$$

is applied to the system. Then the closed-loop system becomes:

$$\begin{aligned} \dot{x}(t) &= (A + BK)x(t) + Ed(t) \\ z(t) &= (C + DK)x(t). \end{aligned} \tag{10.6.6}$$

The transfer function from d to z is:

$$T_{zd}(s) = (C + DK)(sI - A - BK)^{-1}E. \tag{10.6.7}$$

Suppose that the influence of the disturbance vector $d(t)$ on the output $z(t)$ is measured by the H_∞ -norm of $T_{zd}(s)$. The goal of the state feedback H_∞ control problem is to find a constant feedback matrix K such that the closed-loop system is stable and the H_∞ -norm of the transfer matrix $T_{zd}(s)$ is less than a prescribed tolerance.

Specifically, the state feedback H_∞ problem is stated as follows:

Given a positive real number γ , find a real $m \times n$ matrix K such that the closed-loop system is stable and that $\|T_{zd}(s)\|_\infty < \gamma$.

Thus by solving the above problem, one will stabilize perturbed versions of the original system, as long as the size of the perturbations does not exceed a certain given tolerance.

The following theorem due to Doyle, Glover, Khargonekar, and Francis (1989) states a solution of the above problem in terms of the solution of an ARE.

Theorem 10.6.4 (A State Feedback H_∞ Theorem) *Let the pair (A, C) be observable and the pairs (A, B) , and (A, E) be stabilizable. Assume that $D^T D = I$, and $D^T C = 0$. Then the H_∞ control problem (as stated above) has a solution if there exists a positive semi-definite solution X of the algebraic Riccati equation*

$$A^T X + X A - X \left(B B^T - \frac{1}{\gamma^2} E E^T \right) X + C^T C = 0, \quad (10.6.8)$$

such that $A + \left(\frac{1}{\gamma^2} E E^T - B B^T \right) X$ is stable. In this case one such state feedback matrix K is given by

$$K = -B^T X. \quad (10.6.9)$$

Proof: The proof follows by noting the relationship between the algebraic Riccati equation (10.6.8) and the associated Hamilton matrix

$$H_\gamma = \begin{pmatrix} A & \frac{1}{\gamma^2} E E^T - B B^T \\ -C^T C & -A^T \end{pmatrix}, \quad (10.6.10)$$

as stated in Theorem 10.5.2, and then, applying Theorem 10.6.1 to the transfer function matrix $T_{zd}(s)$. ■

Notes:

- (i) The application of Theorem 10.6.1 to $T_{zd}(s)$ amounts to replacing A, B, C , and R of Theorem 10.6.1 as follows:

$$\begin{aligned} A &\rightarrow A + BK = A - BB^T X \\ C &\rightarrow C + DK = C - DB^T X \\ B &\rightarrow E \\ R &\rightarrow \gamma^2 I - I = (\gamma^2 - 1)I, \end{aligned}$$

and using the assumptions $D^T D = I$ and $D^T C = 0$.

- (ii) The Riccati equation (10.6.8) is not the standard LQ Riccati equation, the CARE (Equation (10.5.2)), because the term $(BB^T - \frac{1}{\gamma^2}EE^T)$ may be indefinite for certain values of γ .

However, when $\gamma \rightarrow \infty$, the Riccati equation (10.6.8) becomes the CARE with $R = I$:

$$XA + A^T X - XBB^T X + C^T C = 0$$

- (iii) It can be shown [Exercise 26] that if H_γ has imaginary eigenvalues, then the H_∞ problem as defined above does not have a solution.

In a more realistic situation when a dynamic measurement feedback is used, rather than the state feedback as used here, the solution of the corresponding H_∞ control problem is provided by a **pair of algebraic Riccati equations**. Details can be found in the pioneering paper of Doyle, Glover, Khargonekar, and Francis (1989), and in the recent books by Kimura (1996), Green and Limebeer (1995), Zhou, Doyle and Glover (1996). We only state the result from the paper of Doyle et al. (1989).

10.6.3 The H_∞ Control Problem: Output Feedback Case

Consider a system described by the state-space equations

$$\begin{aligned}\dot{x}(t) &= Ax(t) + B_1w(t) + B_2u(t) \\ z(t) &= C_1x(t) + D_{12}u(t) \\ y(t) &= C_2x(t) + D_{21}w(t)\end{aligned}\tag{10.6.11}$$

where

- $x(t)$ – the state vector
- $w(t)$ – the disturbance signal
- $u(t)$ – the control input
- $z(t)$ – the controlled output
- $y(t)$ – the measured output.

The transfer function from the inputs $\begin{bmatrix} w \\ u \end{bmatrix}$ to the outputs $\begin{bmatrix} z \\ y \end{bmatrix}$ is given by

$$G(s) = \begin{pmatrix} 0 & D_{12} \\ D_{21} & 0 \end{pmatrix} + \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} (sI - A)^{-1} (B_1, B_2) = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}.$$

Define a feedback controller $K(s)$ by $u = K(s)y$.

Then the closed-loop transfer function matrix $T_{zw}(s)$ from the disturbance w to the output z is given by

$$T_{zw}(s) = G_{11} + G_{12}K(I - G_{22}K)^{-1}G_{21}.$$

Then the goal of the output feedback H_∞ control problem in this case is to find a controller $K(s)$ that $\|T_{zw}(s)\|_\infty < \gamma$, for a given positive number γ .

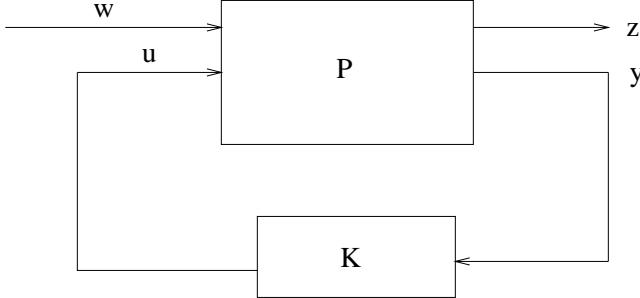


Figure 10.5 Output Feedback H_∞ Configuration

In the above configuration, P is the linear system described by 10.6.11.

A solution of the above H_∞ control problem is given in the following theorem. The following **assumptions** are made:

$$(i) (A, B_1) \text{ is stabilizable and } (A, C_1) \text{ is detectable.} \quad (10.6.12)$$

$$(ii) (A, B_2) \text{ is stabilizable and } (A, C_2) \text{ is detectable.} \quad (10.6.13)$$

$$(iii) D_{12}^T(C_1, D_{12}) = (0, I) \quad (10.6.14)$$

$$(iv) \begin{pmatrix} B_1 \\ D_{21} \end{pmatrix} D_{21}^T = \begin{pmatrix} 0 \\ I \end{pmatrix}. \quad (10.6.15)$$

Here I stands for an identity matrix of appropriate dimension.

Theorem 10.6.5 (An Output Feedback H_∞ Theorem)

Under the assumptions (10.6.12-10.6.15), the output feedback H_∞ control problem as stated above has a solution if there exist unique symmetric positive semidefinite stabilizing solutions X and Y , respectively, to the pair of algebraic Riccati equations

$$XA + A^T X - X \left(B_2 B_2^T - \frac{1}{\gamma^2} B_1 B_1^T \right) X + C_1^T C_1 = 0 \quad (10.6.16)$$

$$AY + Y A^T - Y \left(C_2^T C_2 - \frac{1}{\gamma^2} C_1^T C_1 \right) Y + B_1 B_1^T = 0 \quad (10.6.17)$$

and $\rho(XY) < \gamma^2$, where $\rho(XY)$ is the spectral radius of the matrix XY .

Furthermore, in this case, one such controller is given by the transfer function

$$K(s) = -F(sI - \hat{A})^{-1}ZL, \quad (10.6.18)$$

where

$$\hat{A} = A + \frac{1}{\gamma^2}B_1B_1^TX + B_2F + ZLC_2 \quad (10.6.19)$$

and

$$F = -B_2^TX, \quad L = -YC_2^T, \quad Z = (I - \frac{1}{\gamma^2}YX)^{-1} \quad (10.6.20)$$

Proof: For a proof of Theorem 10.6.5, we refer the readers to the original paper of Doyle et al. (1989). ■

Notes. (1) The second Riccati equation is dual to the first one and is of the type that arises in the Kalman Filter (**Chapter 12**).

(2) A general solution without the assumptions (10.6.14) and (10.6.15) can be found in Glover and Doyle (1988).

MATLAB Note: MATLAB function **care** cannot be used to solve the Riccati equations (10.6.16-10.6.17) arising in the output feedback H_∞ control problem, because the matrices $(B_2B_2^T - \frac{1}{\gamma^2}B_1B_1^T)$ and $(C_2^TC_2 - \frac{1}{\gamma^2}C_1^TC_1)$ might be indefinite. However, the function **hinf** from the **Robust Control Toolbox** implements the output feedback H_∞ control problem.

Example 10.6.3 (Zhou, Doyle, and Glover (1996), 440-443).

Let

$$A = a, \quad B_1 = (1, 0), \quad B_2 = b_2$$

$$C_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad D_{12} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$C_2 = c_2, \quad D_{21} = (0, 1).$$

$$\text{Then } D_{12}^T(C_1, D_{12}) = (0, 1) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = (0, 1), \quad \text{and } \begin{pmatrix} B_1 \\ D_{21} \end{pmatrix} D_{21}^T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Thus the conditions (10.6.14) and (10.6.15) are satisfied.

Let $a = -1, b_2 = c_2 = 1$. Let $\gamma = 2$.

Then it is easy to see that

$$\rho(XY) < \gamma^2,$$

$$T_{zw} = \begin{pmatrix} -1.7321 & 1 & -0.7321 \\ 1 & 0 & 0 \\ -0.7321 & 0 & -0.7321 \end{pmatrix},$$

and $\| T_{zw} \|_\infty = 0.7321 < \gamma = 2$.

Optimal H_∞ Control.

The output H_∞ -control problem in this section is usually known as **Suboptimal H_∞ Control** problem.

Ideally, we should have considered **Optimal H_∞ Control problem:**

Find all admissible controllers $K(s)$ such that $\| T_{zw} \|_\infty$ is minimized.

Finding an optimal H_∞ -controller is demanding both theoretically and computationally and in practice, very often a suboptimal controller is enough, because suboptimal controllers are close to the optimal ones.

The behavior of the suboptimal H_∞ -controller solution as γ approaches the infimal achievable norm, denoted by γ_{opt} , is discussed in the book by Zhou, Doyle and Glover (1996, pp. 438-443). It is shown there that for Example 10.6.2, $\gamma_{opt} = \| T_{zw} \|_\infty = 0.7321$.

10.7 The Complex Stability Radius and Riccati Equation

Assume in this section that A , B , and C are **complex matrices**. In Chapter 7, we introduced the concept of stability radius in the context of robust stabilization of the stable system $\dot{x} = Ax$ under structured perturbations of the form $B\Delta C$. The system

$$\dot{x} = (A + B\Delta C)x \quad (10.7.1)$$

may be interpreted as a closed-loop system obtained by applying the output feedback (with unknown feedback matrix Δ) to the system (10.7.3) given below. Thus the stability radius is related to the output feedback stabilization, as well.

In this section, we will discuss the role of the complex stability radius $r_C(A, B, C)$ in certain parametric optimization problems.

Consider the following parametric optimization problem: Minimize

$$J_\rho(x) = \int_0^\infty [\| u(t) \|^2 - \rho \| y(t) \|^2] dt \quad (10.7.2)$$

subject to

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx. \end{aligned} \quad (10.7.3)$$

If $\rho \leq 0$, then above is the usual *LQR* problem, and therefore, can be solved by solving the associated Riccati equation, as we have just seen. We will now show that for certain other values of $\rho > 0$, the above optimization problem is still solvable, by relating ρ to the stability radius. The key to that is to show the existence of a stabilizing solution of the associated Riccati equation for a given value of ρ .

Before we state the main result, we note the following, that shows that for certain values of ρ , the minimization cost is finite. **For simplicity, we will write $r_C(A, B, C)$ as r .**

Theorem 10.7.1 Let $J_\rho(x)$ be defined by (10.7.2). Then

- (i) $\inf J_\rho(0) = 0$ if and only if $\rho \leq r^2$ if and only if $I - \rho G^*(i\omega)G(i\omega) \geq 0$ for all $\omega \in \mathbb{R}$
- (ii) Suppose A is stable, and $r < \infty$. Then for all $\rho \in (-\infty, r^2)$ we have $|\inf J_\rho(x_0)| < \infty$.

Proof: See Hinrichsen and Pritchard (1986a). ■

The algebraic Riccati equation associated with the above minimization problem is

$$XA + A^*X - \rho C^*C - XBB^*X = 0 \quad (10.7.4)$$

Since this equation is dependent on ρ , we denote this Riccati equation by ARE_ρ .

The parametrized Hamiltonian matrix associated with the ARE_ρ is

$$H_\rho = \begin{pmatrix} A & -BB^* \\ \rho C^*C & -A^* \end{pmatrix} \quad (10.7.5)$$

The following Theorems characterize $r_{\mathbb{C}} (= r)$ in terms of H_ρ .

Theorem 10.7.2 (Characterization of the Complex Stability Radius)

Let H_ρ be defined by (10.7.5). Then $\rho < r$ if and only if H_{ρ^2} does not have an eigenvalue on the imaginary axis.

Proof: See Hinrichsen and Pritchard (1986a).

Example 10.7.1 Consider Example 7.8.1.

$$A = \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \quad C = (1, 0).$$

From Example 7.8.1, we know that $r^2 = \frac{3}{4}$.

Case 1. Let $\rho = 0.5 < r_{\mathbb{C}} = r = 0.8660$. Then $H_{\rho^2} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & -1 \\ 0.25 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$.

The eigenvalues of H_{ρ^2} are $-0.4278 \pm 0.8264j, 0.4278 \pm 0.8264j$. Thus H_{ρ^2} does not have a purely imaginary eigenvalue.

Case 2. Let $\rho = 1 > r_{\mathbb{C}} = r = 0.8660$

$$H_{\rho^2} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & -1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

The eigenvalues of H_{ρ^2} are $0.0000 \pm 1.0000j, 0, 0$, which are purely imaginary.

Therefore we obtain an upper and a lower bound for $r : 0.5 < r \leq 1$.

We have already mentioned the relationship between a Hamiltonian matrix and the associated algebraic Riccati equation. In view of Theorem 10.7.2, therefore, the following result is not surprising. The proof of Theorem 10.7.3 has been taken from Hinrichsen and Pritchard (1986b).

Theorem 10.7.3 (Stability Radius and Algebraic Riccati Equation) *Let A be a complex stable matrix and let $r \equiv r_{\mathbb{C}}(A, B, C) < \infty$. Let $\rho \in (-\infty, r^2)$. Then there exists a unique Hermitian stabilizing solution X of the Riccati equation (10.7.4):*

$$XA + A^*X - \rho C^*C - XBB^*X = 0.$$

Moreover, when $\rho = r^2$, there exists a unique solution X having the property that the matrix $A - BB^*X$ is unstable.

Conversely, if A is stable and if there exists a Hermitian solution X of the above algebraic Riccati equation, then necessarily $\rho \leq r^2$.

Remark: Note that if the Riccati equation (10.7.4) has a stabilizing solution X , then the control law

$$u(t) = -B^*Xx(t)$$

minimizes the performance index (10.7.2), and the minimum value of the performance index is $x_0^T X x_0$.

Note: There is no controllability assumption here on the pair (A, B) .

Proof: Let T be a nonsingular matrix such that

$$TAT^{-1} = \begin{pmatrix} A_1 & A_2 \\ 0 & A_3 \end{pmatrix}, \quad TB = \begin{pmatrix} B_1 \\ 0 \end{pmatrix},$$

and

$$CT^{-1} = (C_1, C_2)$$

where (A_1, B_1) is controllable. Multiplying the Riccati equation (10.7.4) on the left by T^{-1*} and on the right by T^{-1} and setting $T^{-1*}XT^{-1} = \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix}$,

we have

$$\begin{aligned} \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} \begin{pmatrix} A_1 & A_2 \\ 0 & A_3 \end{pmatrix} + \begin{pmatrix} A_1^* & 0 \\ A_2^* & A_3^* \end{pmatrix} \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} - \rho \begin{pmatrix} C_1^*C_1 & C_1^*C_2 \\ C_2^*C_1 & C_2^*C_2 \end{pmatrix} \\ - \begin{pmatrix} X_1B_1B_1^*X_1 & X_1B_1B_1^*X_2 \\ X_3B_1B_1^*X_1 & X_3B_1B_1^*X_2 \end{pmatrix} = 0 \end{aligned} \tag{10.7.6}$$

The equation (10.7.6) breaks into the following four equations:

$$X_1A_1 + A_1^*X_1 - \rho C_1^*C_1 - X_1B_1B_1^*X_1 = 0 \tag{10.7.7}$$

$$X_2 A_3 + (A_1 - B_1 B_1^* X_1^*)^* X_2 + X_1 A_2 - \rho C_1^* C_2 = 0 \quad (10.7.8)$$

$$X_3 (A_1 - B_1 B_1^* X_1) + A_3^* X_3 + A_2^* X_1 - \rho C_2^* C_1 = 0 \quad (10.7.9)$$

$$X_4 A_3 + A_3^* X_4 + X_3 A_2 + A_2^* X_2 - \rho C_2^* C_2 - X_3 B_1 B_1^* X_2 = 0 \quad (10.7.10)$$

Since (A_1, B_1) is controllable, there is a unique solution $X_{1\rho}$ of (10.7.7) with the property that $A_1 - B_1 B_1^* X_{1\rho}$ is stable if $\rho \in (-\infty, r^2)$, and if $\rho = r^2$, then $A_1 - B_1 B_1^* X_{1\rho}$ is not stable. (In fact it has one eigenvalue on the imaginary axis). Substituting this stabilizing solution $X_{1\rho}$ into the Sylvester equations (10.7.8), (10.7.9), it is seen that the solutions $X_{2\rho}$ and $X_{3\rho}$ of (10.7.8), (10.7.9) are unique and moreover $X_{3\rho} = X_{2\rho}^*$. Substituting these $X_{2\rho}$ and $X_{3\rho}$ in the last equation (10.7.10), we see that $X_{4\rho}$ is also unique and $X_{4\rho}^* = X_{4\rho}$. Finally, we note that the matrix $TAT^{-1} - TB \cdot B^* T^* X_\rho$, where $X_\rho = \begin{pmatrix} X_{1\rho} & X_{2\rho} \\ X_{3\rho} & X_{4\rho} \end{pmatrix}$, is stable. Thus $A - BB^* X_\rho$ is stable.

To prove of the converse, we note that if $X = X^*$ satisfies the Riccati equation (10.7.6), then

$$(A - j\omega I)^* X + X(A - j\omega I) - \rho C^* C - XBB^* X = 0$$

for all $\omega \in \mathbb{R}$. Thus

$$\begin{aligned} 0 &\leq (B^* X(A - j\omega I)^{-1} B - I)^* (B^* X(A - j\omega I)^{-1} B - I) \\ &= I - \rho G^*(j\omega) G(j\omega) \text{ for all } \omega \in \mathbb{R}. \end{aligned}$$

Thus $\rho \leq r^2$ by the first part of Theorem 10.7.1. ■

Example 10.7.2

$$A = \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}, \quad B = (0, -1)^T, \quad C = (1, 0)$$

Then we know from Example 7.8.1 that $r^2 = \frac{3}{4}$.

Choose $\rho = \frac{1}{2}$.

Then the unique solution X_ρ to the Riccati equation

$$XA + A^T X - XBB^T X - \rho C^T C = 0$$

is $X_\rho = \begin{pmatrix} -0.5449 & -0.2929 \\ -0.2929 & -0.3564 \end{pmatrix}$, which is symmetric.

The eigenvalues of $A - BB^T X_\rho$ are $-0.3218 \pm 0.7769j$. So, the matrix $A - BB^T X_\rho$ is stable. Thus X_ρ is a stabilizing solution.

If ρ is chosen to $\frac{3}{4}$, then the solution $X = \begin{pmatrix} -1 & -0.5 \\ -0.5 & -1 \end{pmatrix}$, which is symmetric, but not stabilizing. The eigenvalues of $A - BB^T X$ are $0 \pm 0.7071j$.

A Bisection Method for Computing the Complex Stability Radius

Theorem 10.7.2 suggests a bisection algorithm for computing the complex stability radius $r_{\mathbb{C}}$. The idea is to determine $r_{\mathbb{C}}$ as that value of ρ for which the Hamiltonian matrix H_ρ given by (10.7.5), associated with the Riccati equation (10.7.4), has an eigenvalue on the imaginary axis for the first time.

If ρ_0^- and ρ_0^+ are some lower and upper estimates of $r_{\mathbb{C}}$, then the successive better estimates can be obtained by the following algorithm.

Algorithm 10.7.1 A Bisection Method for the Complex Stability Radius.

Inputs:

1. The system matrices A, B , and C .
2. Some upper and lower estimates ρ_0^+ and ρ_0^- of the complex stability radius ρ .

Output: An approximate value of the complex stability radius ρ .

For $k = 0, 1, 2, \dots$, do until convergence

Step 1. Take $\rho_k = \frac{\rho_k^- + \rho_k^+}{2}$ and compute $H_{\rho_k^2}$.

Step 2. If $H_{\rho_k^2}$ has eigenvalues on the imaginary axis, set $\rho_{k+1}^- \equiv \rho_k^-$ and $\rho_{k+1}^+ \equiv \rho_k$.

Otherwise set $\rho_{k+1}^- \equiv \rho_k$ and $\rho_{k+1}^+ \equiv \rho_k^+$.

End

Example 10.7.3 Consider Example 10.7.1. Take $\rho_0^- = 0$, $\rho_0^+ = 1$.

$k=0$. **Step 1.** $\rho_0 = \frac{1}{2}$. $H_{\rho_0^2}$ does not have an imaginary eigenvalue.

Step 2. $\rho_1^- = \frac{1}{2}$, $\rho_1^+ = 1$.

$k=1$. **Step 1.** $\rho_1 = \frac{3}{4}$. $H_{\rho_1^2}$ does not have an imaginary eigenvalue.

Step 2. $\rho_2^- = \frac{3}{4}$, $\rho_2^+ = 1$

$k=2$. **Step 1.** $\rho_2 = \frac{7}{8}$. $H_{\rho_2^2}$ has an imaginary eigenvalue.

Step 2. $\rho_3^- = \frac{3}{4}$, $\rho_3^+ = \frac{7}{8}$

$k=3$. **Step 1.** $\rho_3 = \frac{13}{16}$. $H_{\rho_3^2}$ does not have an imaginary eigenvalue.

Step 2. $\rho_4^- = \frac{13}{16}$, $\rho_4^+ = \frac{7}{8}$

$k=4$. $\rho_4 = \frac{27}{32}$.

The iteration is converging towards $r = 0.8660$. The readers are asked to verify this by carrying out some more iterations.

MATCONTROL NOTE: Algorithm 10.7.1 has been implemented in MATCONTROL function **stabradc**.

10.8 Some Selected Software

10.8.1 MATLAB CONTROL SYSTEM TOOLBOX

LQG design tools.

lqr	- LQ-optimal gain for continuous systems.
dlqr	- LQ-optimal gain for discrete systems.
lqry	- LQ regulator with output weighting.
lqrds	- Discrete LQ regulator for continuous plant.
care	- Solve continuous-time algebraic Riccati equations.
dare	- Solve discrete-time algebraic Riccati equations.
norm	- H_2 and H_∞ - norms.

10.8.2 MATCONTROL

STABLYAPC - Feedback stabilization of continuous-time system using Lyapunov equation

STABLYAPD - Feedback stabilization of discrete-time system using Lyapunov equation

STABRADC - Finding the complex stability radius using the bisection method

HINFNRM - Computing H_∞ -norm using the bisection method

10.8.3 CSP-ANM

Feedback stabilization

- Constrained feedback stabilization is computed by **StateFeedbackGains** [*system, region*], where the available regions are **DampingFactorRegion** [α], **SettlingTimeRegion** [t_s, ϵ], **DampingRatioRegion** [ζ_{\min}] and **NaturalFrequencyRegion** [$\omega_{n\min}$], and their intersections.
- The Lyapunov algorithms for the feedback stabilization is implemented as **StateFeedbackGains** [*system, region, Method* → **LyapunovShift**] and **StateFeedbackGains** [*system, region, Method* → **PartialLyapunovShift**].

10.8.4 SLICOT

Optimal Regulator Problems, System Norms and Complex Stability Radius

SB10DD	H -infinity (sub)optimal state controller for a discrete-time system
SB10FD	H -infinity (sub)optimal state controller for a continuous-time system
AB13BD	H_2 or L_2 norm of a system
AB13CD	H -infinity norm of a continuous-time stable system
AB13ED	Complex stability radius, using bisection
AB13FD	Complex stability radius, using bisection and SVD.

10.8.5 \mathbf{MATRIX}_X

Purpose: Computing L -infinity norm of the transfer matrix of a discrete-time system.

Syntax: [SIGMA, OMEGA]=DLINFNORM (S, NS, { TOL, { MAXITER } })

Purpose: Compute optimal state feedback gain for discrete-time system.

Syntax: [EVAL, KR]=DREGULATOR (A, B, RXX, RUU, RXU) OR
[EVAL, KR, P]=DREGULATOR (A, B, RXX, RUU, RXU)

Purpose: Computing L -infinity norm of a transfer matrix

Syntax: [SIGMA, OMEGA]=LINFNORM (S, NS, { TOL, { MAXITER } })

Purpose: Compute optimal state feedback gain for continuous-time system.

Syntax: [EVAL, KR]=REGULATOR (A, B, RXX, RUU, RXU) OR
[EVAL, KR, P]=REGULATOR (A, B, RXX, RUU, RXU)

Purpose: Computes and plots the Singular Values of a continuous system

Syntax: [OMEGA, SVALS]=SVPLOT (S, NS, WMIN, WMAX, { NPTS} , { OPTIONS })
OR
[SVALS]=SVPLOT (S, NS, OMEGA, { OPTIONS })

10.9 Summary and Review

The following topics have been discussed in this chapter.

- State feedback stabilization
- Eigenvalue and eigenstructure assignments by state and output feedback

- The LQR design
- H_∞ -control problems
- Stability radius

I. Feedback stabilization.

The problem of stabilizing the continuous-time system

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}$$

by using the state feedback law $u(t) = -Kx(t)$ amounts to finding a feedback matrix K such that $A - BK$ is stable.

The state-feedback stabilization of a discrete-time system is similarly defined.

The **characterizations** of the continuous-time and discrete-time state feedback stabilizations are, respectively, given in **Theorem 10.2.1** and **Theorem 10.2.2**.

It is shown how a system can be stabilized using the solution of a Lyapunov equation. For the continuous-time system, the Lyapunov equation to be solved is

$$-(A + \beta I)X + X(-(A + \beta I))^T = -2BB^T,$$

where β is chosen such that $\beta > |\lambda_{\max}(A)|$.

The stabilizing feedback matrix K is given by

$$K = B^T X^{-1}.$$

In the discrete-time case, the Lyapunov equation to be solved is

$$AXA^T - \beta^2 X = 2BB^T,$$

where β is chosen such that $0 < \beta \leq 1$ and $|\lambda| \geq \beta$ for any eigenvalue λ of A . The stabilizing feedback matrix in this case is

$$K = B^T(X^T + BB^T)^{-1}A.$$

II. Detectability.

The detectability of the pair (A, C) is a dual concept of the stabilizability of the pair (A, B) . Characterizations of the continuous-time and discrete-time detectability are, respectively, stated in **Theorem 10.3.1** and **Theorem 10.3.3**.

III. The Eigenvalue Assignment.

For the transient responses to meet certain designer's constraints it is required that the eigenvalues of the closed-loop matrix lie in certain specified regions of the complex plane. This consideration gives rise to the well-known eigenvalue assignment (EVA) problem.

The EVA problem by state feedback is defined as follows:

Given the pair (A, B) and a set Λ of the complex numbers, closed under complex conjugations, find a feedback matrix K such that $A - BK$ has the spectrum Λ .

The conditions of solvability for the EVA problem and the uniqueness of the matrix K are:

There exists a matrix K such that the matrix $A - BK$ has the spectrum Λ for every complex-conjugated set Λ if and only if (A, B) is controllable. The feedback matrix K , when it exists, is unique if and only if the system is a single-input system (**Theorem 10.4.1**).

The constructive proof Theorem 10.4.1 and several related well-known formulas such as the **Bass-Gura formula** and the **Ackermann formula** suggest computational methods for **single-input eigenvalue assignment problem**. Unfortunately, however, these methods are based on the reduction of the pair (A, b) to a controller-companion pair, and **are not numerically effective**. Numerically effective algorithms for EVA are based on the reduction of the pair (A, b) or the pair (A, B) (in the multi-output case) to the controller-Hessenberg pair. **These methods will be described in Chapter 11.**

The eigenvalue assignment problem by output feedback is discussed in Section 10.4.2 and a well-known result on this problem is stated in **Theorem 10.4.2**.

IV. The Eigenstructure Assignment

The eigenvalues of the state matrix A determine the rate at which the system response decays or grows. On the other hand, the eigenvectors determine the shape of the response. Thus, in certain practical applications, it is important that both the eigenvalues and the eigenvectors are assigned. The problem is known as the **eigenstructure assignment problem**. The conditions under which eigenstructure assignment is possible are stated in Theorem 10.4.3 for the state feedback law and in Theorem 10.4.4 for the output feedback law.

V. The Linear Quadratic Regulator (LQR) Problem

The continuous-time LQR problem is the problem of finding a control vector $u(t)$ that minimizes the performance index

$$J_C(x) = \int_0^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)]dt$$

subject to

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), x(0) = x_0 \\ y(t) &= Cx(t), \end{aligned}$$

where Q and R are, respectively, the state-weight and the control-weight matrices.

It is shown in **Theorem 10.5.1** that the continuous-time LQR problem has a solution if (A, B) is stabilizable and (A, Q) is detectable.

The solution is obtained by solving the continuous-time algebraic Riccati equation (CARE)

$$XA + A^T X - XSX + Q = 0,$$

where $S = BR^{-1}B^T$.

The optimal control vector $u^0(t)$ is given by

$$u^0(t) = -R^{-1}B^T X x(t),$$

where X is the unique symmetric positive semidefinite solution of the CARE.

The matrix $K = -R^{-1}TB^TX$ is such that $A - BK$ is stable; that is, X is a stabilizing solution.

The LQR design has the following **guaranteed stability and robustness properties**:

Stability Property. The stable open-loop eigenvalues remain stable and the unstable eigenvalues get reflected across the imaginary axis (when $R = \rho I$ and $\rho \rightarrow \infty$).

Robustness Properties. Using the optimal return difference identity, it can be shown that

$$\sigma_{\min}(I + G_{LQ}(j\omega)) \geq 1$$

and $\sigma_{\min}(I + G_{LQ}(j\omega)^{-1}) \geq \frac{1}{2}$, where $G_{LQ}(s) = K(sI - A)^{-1}B$.

These relations show that the upward and downward gain margins are, respectively, ∞ and 0.5. The phase margin is at least $\pm 60^\circ$.

The discrete-time LQR problem is similarly defined. In this case, the performance index is given by

$$J_D(x) = \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k)$$

The discrete-time algebraic equation (DARE) is

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$

If (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable, then the above Riccati equation (DARE) has a unique symmetric positive semidefinite solution X and the optimal control is $u_k^0 = K x_k$, where

$$K = (R + B^T X B)^{-1} B^T X A.$$

Furthermore, X is a discrete-stabilizing solution; that is, $(A - BK)$ is discrete-stable.

VI. H_∞ Control Problems

The H_∞ control problems is concerned with robust stabilization for unstructured perturbations in the frequency domain.

The goal of a H_∞ control is to determine a controller that guarantees a closed-loop system with an H_∞ -norm bounded by a certain specified quantity γ when such a controller exists.

Both the state feedback and the output feedback H_∞ control problems have been discussed briefly in Sections 10.6.2 and 10.6.3, respectively. Both problems require solutions of certain Riccati equations.

Under the assumptions (10.6.12)-(10.6.15), the solution of the output feedback H_∞ control problem reduces to solving a pair of Riccati equations:

$$\begin{aligned} XA + A^T X - X(B_2 B_2^T - \frac{1}{\gamma^2} B_1 B_1^T)X + C_1^T C_1 &= 0 \\ AY + Y A^T - Y(C_2^T C_2 - \frac{1}{\gamma^2} C_1^T C_1)Y + B_1 B_1^T &= 0, \end{aligned}$$

where A, B_1, B_2, C_1 , and C_2 are defined by (10.6.11).

The expression for a H_∞ controller is given in (10.6.18)-(10.6.20).

Also, two computational algorithms: one, the well-known **bisection algorithm** by Boyd et al and the other, the two-step algorithm by Bruinsma et al (1990), for computing the H_∞ -norm are given in Section 10.6.1.

VII. Stability Radius

The concept of stability radius has been defined in Chapter 7. Here a connection of the **complex stability radius** r is made with the algebraic Riccati equation

$$XA + A^* X - \rho C^* C - XBB^* X = 0$$

via the parameterized Hamiltonian matrix

$$H_\rho = \begin{pmatrix} A & -BB^* \\ \rho C^* C & -A^* \end{pmatrix}.$$

It is shown in **Theorem 10.7.3** that if $\rho \in (-\infty, r^2)$, then the above Riccati equation has a unique stabilizing solution X . Conversely, if A is stable and if there exists a Hermitian solution X of the above equation, then $\rho \leq r^2$.

In terms of the eigenvalues of the Hamiltonian matrix H_ρ , it means that $\rho < r$ if and only if H_{ρ^2} does not have an eigenvalue on the imaginary axis (**Theorem 10.7.2**).

Based on the later result, a **bisection algorithm** (**Algorithm 10.7.1**) for determining the complex stability radius is described.

10.10 Chapter Notes and Further Reading

Feedback stabilization and eigenvalue assignment (pole-placement) are two central problems in control theory. For detailed treatment of these problems, see Brockett (1970), Brogan (1991), Friedland (1986), Chen (1984), Kailath (1980), Wonham (1985), etc. Most of the books in linear systems theory, however, do not discuss feedback stabilization via Lyapunov equations. Discussions on feedback stabilization via Lyapunov equations in Section 10.2 have been taken from the papers of Armstrong (1975) and Armstrong and Rublein (1976). For a Schur method for feedback stabilization, see Sima (1981). For stabilization methods of descriptor systems, see Varga (1995). For more on the output feedback problem, see Kimura (1975), Porter (1977), Sridhar and Lindorff (1973), Srinathkumar (1978) and Misra and Patel (1989).

For a discussion on eigenstructure assignment problem, see Andry, Shapiro and Chung (1983). The authoritative book by Anderson and Moore (1990) is an excellent source for a study on the LQR (Linear Quadratic Regulator) problem. The other excellent books on the subject include Athans and Falb (1966), Lewis (1986), Mehrmann (1991), Sima (1996), Kučera (1979), etc. For a proof of the discrete-time LQR Theorem (**Theorem 10.5.3**), see Sage and White (1977). An excellent reference book on theory of Riccati equations is the recent book by Lancaster and Rodman (1995). There are also several nice papers on Riccati equations in the books edited by Bittanti et al. (1991), and Bittanti (1989). H_∞ control problem has been dealt with in details in the books by Kimura (1996), Zhou, Doyle and Glover (1996), Green and Limebeer (1995), Dorato, et al. (1992, 1995). The original papers by Francis and Doyle (1987) and by Doyle, Glover, Khargonekar and Francis (1989) are worth reading. A recent book by Kirsten Morris (2001) contains an excellent coverage on feedback control; in particular, H_∞ feedback control. Algorithm 10.6.1 and Algorithm 10.6.2 for computing the H_∞ -norm have been taken, respectively, from the papers of Boyd et al. (1989) and Bruinsma et al. (1990). A gradient method for computing the optimal H_∞ -norm has been proposed in Pandey et al. (1991). Recently, Lin, Wang and Xu (2000) have proposed numerically reliable algorithms for computing H_∞ -norms of the discrete-time systems, both for the state and the output feedback problems. The discussion on the complex stability radius and Riccati equation in Section 10.7 has been taken from Hinrichsen and Pritchard (1986b). For an iterative algorithm for H_∞ -control by state feedback, see Scherer (1989). Theorem 10.7.3 is an extension of the work of Brockett (1970) and Willems (1971). For an application of the algebraic Riccati equation to compute H_∞ optimization, see Zhou and Kargonekar (1988).

EXERCISES

1. Prove the equivalence of (i) and (iii) in Theorem 10.2.1.
2. Prove Theorem 10.2.2.
3. Prove Theorem 10.2.4.
4. Construct a state-space continuous-time system that is stabilizable, but not controllable.
Apply the Lyapunov stabilization method (modify the method in the book as necessary) to stabilize this system.
5. Repeat Problem 4 with a discrete-time system.
6. Develop algorithms for feedback stabilization, both for the continuous-time and discrete-time systems, based on the reduction of A to the real Schur form (see Sima (1981)).
Compare the efficiency of each of these Schur algorithms with the respective Lyapunov equation based algorithms given in the book.
7. Using the real Schur decomposition of A , develop partial stabilization algorithms, both for the continuous-time and discrete-time systems in which only the unstable eigenvalues of A are stabilized using feedback, leaving the stable eigenvalues unchanged.
8. Prove Theorem 10.3.1.
9. State and prove the discrete counterpart of Theorem 10.3.2.
10. Prove Theorem 10.3.3.
11. Give an alternative proof of the state feedback eigenvalue assignment Theorem (Theorem 10.4.1).
12. Construct an example to verify that if the eigenvalues of the closed-loop system are moved far from those of the open-loop system, a large feedback will be required to place the closed-loop eigenvalues.
13. Using the expression of the transforming matrix T that transforms the system (A, b) to a controller-companion from (10.4.1)-(10.4.2), and the expression for the feedback formula (10.4.5), derive the Bass-Gura formula (10.4.6)
14. Derive the Ackermann formula (10.4.7).
15. Work out an example to illustrate each of the following theorems: Theorem 10.5.1, Theorem 10.5.2, Theorem 10.5.3, Theorem 10.6.1, Theorem 10.6.2, Theorem 10.6.3, Theorem 10.6.4, and Theorem 10.6.5. (Use MATLAB function **care** to solve the associated Riccati equation, whenever needed).

16. (**Design of a regulator with prescribed degree of stability**). Consider the LQR problem of minimizing the cost function

$$J_\alpha = \int_0^\infty e^{\alpha t} (u^T R u + x^T Q x) dt.$$

- (a) Show that the Riccati equation to be solved in this case is:

$$(A + \alpha I)^T X + X(A + \alpha I) + Q - XBR^{-1}B^T X = 0$$

and the optimal control is given by same feedback matrix K as in Theorem 10.5.1.

- (b) Give a physical interpretation of the problem.

17. (**Cross-Weighted LQR**). Consider the LQR problem with the quadratic cost function with a cross penalty on state and control:

$$J_{CW} = \int_0^\infty [x^T Q x + x^T N u + u^T R u] dt$$

subject to $\dot{x} = Ax + Bu$, $x(0) = x_0$, where Q , R , N , are, respectively, the state-weighting matrix, the control weighting matrix and the cross weighting matrix.

Define $A_R = A - BR^{-1}N^T$.

- (a) Show that the Riccati equation to be solved in this case is:

$$XA_R + A_R^T X + (Q - NR^{-1}N^T) - XBR^{-1}B^T X = 0,$$

and the optimal control law is given by $u(t) = -Kx(t)$, where $K = R^{-1}(N^T + B^T X)$.

- (b) What are the assumptions needed for the existence of the unique, symmetric positive-semidefinite solution X of the Riccati equation in (a)?

18. Consider the LQR problem of minimizing the cost

$$J = \int_0^\infty [x^2(t) + \rho^2 u^2(t)] dt,$$

with $\rho > 0$, subject to

$$m\ddot{q} + kq(t) = u(t).$$

- (a) Find an expression for the feedback matrix K in terms of ρ , by solving an appropriate algebraic Riccati equation.
- (b) Plot the closed-loop poles as ρ varies over $0 < \rho < \infty$.
- (c) Write down your observations about the behavior of the closed-loop poles with respect to stability.

19. Using the MATLAB function ***lqr***, design an LQR experiment with a single-input system to study the behavior of the closed-loop poles and the feedback vector with varying ρ in $R = \rho I$ in the range $[1, 10^6]$, taking $\rho = 1, 10, 10^2, 10^3$, and 10^6 . Plot the open loop poles, the closed loop poles and the step responses. Make a table of the gain margins and phase margins with each feedback vector.
20. (a) Using the return-difference identity, show that the i th singular value σ_i of the return-difference matrix with $s = j\omega$ is:

$$\sigma_i(I + G_{LQ}(j\omega)) = [1 + \frac{1}{\rho} \sigma_i^2(H(j\omega))]^{\frac{1}{2}},$$

where $H(s) = C(sI - A)^{-1}B$, $R = \rho I$ and $Q = C^T C$.

- (b) Using the result in (a), prove that

$$\sigma_{min}(I + G_{LQ}(j\omega)) \geq 1.$$

- (c) Using (b), prove that

$$\sigma_{min}(I + (G_{LQ}(j\omega))^{-1}) \geq \frac{1}{2}.$$

21. In certain applications, the homogeneous algebraic Riccati equation:

$$XA + A^T X + XWX = 0$$

is important.

Prove that the above homogeneous Riccati equation has a stabilizing solution (that is, $A + WX$ is stable) if A has no eigenvalues on the imaginary axis.

22. (**Computing the H_∞ -norm over an interval**). Define the H_∞ -norm of $G(s)$ over an interval $0 \leq \alpha < \beta$ as:

$$\| G \|_\infty = \sup \sigma_{max}(C(j\omega))$$

$$\alpha \leq \omega \leq \beta$$

- (a) Develop an algorithm to compute $\| G \|_\infty$ by modifying the bisection algorithm (**Algorithm 10.6.1**) as follows:

1. Take $\gamma_{lb} = \max\{\sigma_{max}(G(j\alpha)), \sigma_{max}(G(j\beta))\}$
2. Modify the eigenvalue test in Step 2 as: if M_γ has no imaginary eigenvalues between $j\alpha$ and $j\beta$.

- (b) Work out a numerical example to test your algorithm.

23. Give a linear algebraic proof of Theorem 10.6.1 (consult the paper by Boyd, Balakrishnan and Kabamba (1989)).

24. Develop an algorithm for computing the H_∞ -norm of a discrete stable transfer function, based on Theorem 10.6.3.
25. (Kimura (1996)). For the second-order system

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= w_1 + u_1 \\ z_1 &= x_1 \\ z_2 &= \delta u_1 \\ y &= c_2 x_1 + d_2 u_2\end{aligned}$$

find the conditions under which the output feedback problem has a solution. Find the transfer function for H_∞ controller.

26. Prove that if the Hamiltonian matrix H_γ defined by (10.6.10) has an imaginary eigenvalue, then the state feedback H_∞ control problem does not have a solution.
27. Prove Theorem 10.6.2: If A is a complex stable matrix, then the distance to instability

$$\beta(A) = \| (sI - A)^{-1} \|_\infty^{-1}.$$

28. (a) Let $G(s) = C(sI - A)^{-1}B$. Then prove

$$r = \begin{cases} \frac{1}{\max_{\omega \in \mathbb{R}} \|G(j\omega)\|}, & \text{if } G \neq 0 \\ \infty, & \text{if } G = 0. \end{cases}$$

(Consult Hinrichsen and Pritchard (1986b)).

- (b) Give an example to show that $r(A, I, I)$ can change substantially under similarity transformation on A .

References

1. J. Ackermann, Der entwurf linear regelung systeme im zustandsraum, *Regulungstechnik und prozessdatenverarbeitung*, vol. 7, pp. 297-300, 1972.
2. B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice Hall, Englewood Cliffs, NJ, 1990.
3. A. N. Andry, Jr., E. Y. Shapiro and J. C. Chung, Eigenstructure assignment for linear systems, *IEEE Trans. Aerospace and Electronic Systems*, vol. AES-19, no. 5, pp.711-729, 1983.
4. E. S. Armstrong, An extension of Bass' algorithm for stabilizing linear continuous constant systems, *IEEE Trans. Automat. Control*, AC-20, pp. 153-154, 1975.
5. E. S. Armstrong and G. T. Rublein, A stabilizing algorithm for linear discrete constant systems, *IEEE Trans. Automat. Control*, AC-21, pp. 629-631, 1976.
6. M. Athans and P. Falb, *Optimal Control*, McGraw Hill, New York, 1966.
7. S.P. Bhattacharyya, H. Chapellat and L. H. Keel, *Robust Control: the Parametric Approach*, Prentice Hall, Information and Systems Sciences Series, Prentice Hall, Upper Saddle River, NJ, 1995, (Thomas Kailath, Editor).
8. S. Bittanti (Ed.), *The Riccati Equation in Control, Systems, and Signals*, Lecture Notes, Pitagora Editrice, Bologna, 1989.
9. S. Bittanti, A.J. Laub and J.C. Willems (Editors), *The Riccati Equation*, Springer-Verlag, Berlin, 1991.
10. S. Boyd, V. Balakrishnan and P. Kabamba, A bisection method for computing the H_∞ -norm of a transfer function matrix and related problems, *Math. Control and Signals Systems*, vol. 2, pp.207-219, 1989.
11. S. Boyd and C.H. Barratt, *Linear Controller Design–Limits of Performance*, Prentice Hall Information and System Sciences Series, Englewood Cliffs, NJ, 1991. (Thomas Kailath, Editor).
12. S. Boyd and V. Balakrishnan, A regularity result for the singular values of a transfer function matrix and a quadratically convergent algorithm for computing its L_∞ -norm, *Syst. Contr. Lett.*, pp. 1-7, 1990.
13. R.W. Brockett, *Finite Dimensional Linear Systems*, Wiley, New York, 1970.
14. W.L. Brogan, *Modern Control Theory*. 3rd Edition, Prentice Hall, Englewood Cliffs, NJ, 1991.

15. N.A. Bruinsma and M. Steinbuch, A fast algorithm to compute the H_∞ -norm of a transfer function matrix, *Syst. Contr. Lett.*, vol. 14, pp.287-293, 1990.
16. C.-T. Chen, *Linear System Theory and Design*, CBS College Publishing, New York, 1984.
17. P. Dorato, L. Fortuna and G. Muscato, *Robust control for unstructured perturbations—An introduction*, Springer-Verlag, Berlin, 1992, (M. Thoma and A. Wyner, Editors).
18. P. Dorato, C. Abdallah and V. Cerone, *Linear Quadratic Control*, An Introduction, Prentice Hall, Englewood Cliffs, NJ, 1995.
19. J.C. Doyle, B. Francis and A. Tannenbaum, *Feedback Control Theory*, McMillan Publishing Company, New York, 1992.
20. J. Doyle, K. Glover, P. Khargonekar and B. Francis, State-space solutions to standard H_2 and H_∞ control problems, *IEEE Trans. Automat Control*, AC-34, pp.831-847, 1989.
21. D.F. Enns, Model reduction with balanced realizations: An error bound and a frequency weighted generalization, *Proc. IEEE Conf. Dec. Control*, pp.127-132, 1984.
22. B.A. Francis and J.C. Doyle, Linear Control Theory with an H_∞ optimality criterion, *SIAM J. Contr. optimiz.*, vol. 25, pp.815-844, 1987.
23. B.A. Francis, *A Course in H_∞ Control Theory*, Lecture Notes in Control and Information Sciences, vol. 88, 1987.
24. G.F. Franklin, J.D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Addison-Wesley, New York, 1986.
25. B. Friedland, *Control System Design: An Introduction to State-Space Methods*, McGraw-Hill, New York, 1986.
26. K. Glover, All optimal Hankel-norm approximations of linear multivariable systems and their L_∞ -error bounds, *Int. J. Control*, vol. 39, pp.1115-1193, 1984.
27. K. Glover and J. Doyle, State-space formulae for all stabilizing controllers that satisfy an H_∞ -norm bound and relations to risk sensitivity, *Syst. Contr. Lett.*, vol. 11, pp. 167-172, 1988.
28. M. Green and D.J.N. Limebeer, *Linear Robust Control*, Prentice Hall Information and System Sciences Series, Prentice Hall, Englewood Cliffs, NJ, 1995, (Thomas Kailath, Editor).
29. D. Hinrichsen and A.J. Pritchard, Stability radii of linear systems, *Syst. Contr. Lett.*, vol. 7, pp.1-10, 1986a.
30. D. Hinrichsen and A.J. Pritchard, Stability radius for structured perturbations and the algebraic Riccati equation, *Syst. Contr. Lett.*, vol. 8, pp.105-113, 1986b.

31. T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.
32. H. Kimura, Pole assignment by gain output feedback, *IEEE Trans. Automat. Control*, AC-20, no. 4, pp.509-516, 1975.
33. H. Kimura, *Chain-Scattering to H^∞ Control*, Birkhäuser, Boston, 1996.
34. V. Kucera, *Discrete Linear Control*, John Wiley & Sons, New York, 1979.
35. H. Kwakernaak and R. Silvan, *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.
36. P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, UK, 1995.
37. William S. Levine (Editor), *The Control Handbook*, CRC Press and IEEE Press, Boca Raton, Fl., 1996.
38. F.L. Lewis, *Optimal Control*, John Wiley & Sons, New York, 1986.
39. F.L. Lewis, *Applied Optimal Control and Estimation – Digital Design and Implementation*, Prentice Hall and Texas Instruments Digital Signal Processing Series, Prentice Hall, Englewood Cliffs, NJ, 1992.
40. W.-W. Lin, J.-S. Wang, and Q.-F. Xu, on the Computation of the optimal H_∞ -norms for two feedback control problems, *Lin. Alg. Appl.*, vol. 287, pp. 233-255, 1999.
41. W.-W. Lin, J.-S. Wang, and Q.-F. Xu, Numerical computation of the minimal H_∞ -norms of the discrete-time state and output feedback control problems, *SIAM. J. Num. Anal.*, vol. 38, no. 2, pp. 515-547, 2000.
42. J.M. Maciejowski, *Multivariable Feedback Design*, Addison-Wesley, Wokingham, 1989.
43. V. Mehrmann, *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*, Lecture Notes in Control and Information Sciences, Springer-Verlag, Heidelberg, 1991.
44. K. Morris, *Introduction to Feedback Control*, Academic Press, Boston, 2001.
45. P. Misra and R.V. Patel, Numerical algorithms for eigenvalue assignment by constant and dynamic output feedback, *IEEE Trans. Automat. Control*, vol. 34, pp.579-588, 1989.
46. B.C. Moore, On the flexibility offered by state feedback in multivariable systems beyond closed loop eigenvalue assignment, *IEEE Trans. Automat. Control*, vol. AC-21, pp.689-692, 1976.
47. P. Pandey, C.S. Kenney, A. Packard and A.J. Laub, A gradient method for computing the optimal H_∞ norm, *IEEE Trans. Automat. Control*, vol. 36, pp.887-890, 1991.

48. B. Porter, Eigenvalue assignment in linear multivariable systems by output feedback, *Int. J. Control.*, vol. 25, pp.483-490, 1977.
49. G. Robel, On Computing the infinity norm, *IEEE Trans. Autoimat. Control*, vol. 34, pp.882-884, 1989.
50. M.G. Safonov and M. Athans, Gain and phase margins of multiloop *LQG* regulators, *IEEE Trans. Automat. Control*, AC-22, pp. 173-179, 1977.
51. M.G. Safonov, A.J. Laub, and G.L. Hartmann, Feedback properties of multivariable systems: The role and use of the return difference matrix, *IEEE Trans. Automat. Control*, vol. AC-26, pp. 47-65, 1981.
52. M.G. Safonov, *Stability and Robustness of Multivariable Feedback Systems*, MIT Press, Boston, 1980.
53. A.P. Sage and C.C. White, *Optimal Systems Control*, 2nd Edition, Prentice Hall, Englewood Cliffs, NJ, 1977.
54. C. Scherer, H_∞ -control by state feedback: An iterative algorithm and characterization of high-gain occurrence, *Syst. Contr. Lett.*, vol. 12, pp.383-391, 1989.
55. V. Sima, An efficient Schur method to solve the stabilizing problem, *IEEE Trans. Automat. Control*, vol. AC-26, pp.724-725, 1981.
56. V. Sima, *Algorithms for Linear-Quadratic Optimization*, Marcel Dekker, New York, 1996.
57. B. Sridhar and D.P. Lindorff, Pole placement with constant gain output feedback, *Int. J. Control.*, vol. 18, pp.993-1003, 1973.
58. S. Srinathkumar, Eigenvalue/eigenvector assignment using output feedback, *IEEE Trans. Automat. Control*, vol. AC-21, pp.79-81, 1978.
59. A. Varga, On stabilization methods of descriptor systems, *Syst. Contr. Lett.*, vol. 24, pp.133-138, 1995.
60. J.C. Willems, Least squares stationary optimal control and the algebraic Riccati equation, *IEEE Trans. Automat. Control*, AC-16, pp.621-634, 1971.
61. W.H. Wonham, *Linear Multivariable Control: A geometric approach*, Springer-Verlag, Berlin, 1985.
62. G. Zames, Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses, *IEEE Trans. Automat. Control*, AC-26, pp. 301-320, 1981.
63. K. Zhou, J.C. Doyle and K. Glover, *Robust and Optimal Control*, Prentice Hall, Upper Saddle River, NJ., 1996.

64. K. Zhou and P. Khargonekar, An Algebraic Riccati equation approach to H_∞ optimization, *Syst. Contr. Lett.*, vol. 11, no. 2, pp. 85-91, 1988.

Chapter 11

NUMERICAL METHODS AND CONDITIONING OF THE EIGENVALUE ASSIGNMENT PROBLEMS

Contents

11.1 Introduction	456
11.2 Numerical Methods for the Single-input Eigenvalue Assignment Problem	458
11.2.1 A Recursive Algorithm for the Single-input EVA Problem	461
11.2.2 An Error Analysis of the Recursive Single-Input Method	466
11.2.3 The QR and RQ Implementations of Algorithm 11.2.1	467
11.2.4 Explicit and Implicit RQ Algorithms	473
11.3 Numerical Methods for the Multi-input Eigenvalue Assignment Problem	473
11.3.1 A Recursive Multi-input Eigenvalue Assignment Algorithm	474
11.3.2 The Explicit <i>QR</i> Algorithm for the Multi-input EVA Problem	477
11.3.3 The Schur Method for the Multi-input Eigenvalue Assignment Problem	484
11.4 Conditioning of the Feedback Problem	493
11.4.1 The Single-Input Case	493
11.4.2 The Multi-input Case	496
11.4.3 Absolute and Relative Condition Numbers	497
11.5 Conditioning of the Closed-loop Eigenvalues	499
11.6 Robust Eigenvalue Assignment	501
11.6.1 Measures of Sensitivity	501

11.6.2 Statement of the Robust Eigenvalue Assignment Problem	502
11.6.3 A Solution Technique for the Robust Eigenvalue Assignment Problem	503
11.7 Table of Comparison for Efficiency and Stability: Single-input EVA Problem	509
11.8 Table of Comparison for Efficiency and Stability: Multi-input EVA Problem	510
11.9 Comparative Discussion of Various Methods and Recommendation	511
11.10 Some Selected Software	512
11.10.1 MATLAB CONTROL SYSTEM TOOLBOX	512
11.10.2 MATCONTROL	512
11.10.3 CSP-ANM	512
11.10.4 SLICOT	513
11.10.5 MATRIX _X	513
11.10.6 POLEPACK	513
11.11 Summary and Review	513
11.12 Chapter Notes and Further Reading	516

Topics covered

- Numerical Methods for the Single-input and Multi-input Eigenvalue Assignment Problems
- Sensitivity Analyses of the Feedback and Eigenvalue Assignment Problems and Conditioning of the Closed-loop Eigenvalues
- Robust Eigenvalue Assignment

11.1 Introduction

We have introduced the eigenvalue assignment (EVA) problem (**pole-placement problem**) in Chapter 10 and given the results on existence and uniqueness of the solution.

In this chapter, we study numerical methods and the perturbation analysis for this problem.

There are many methods for the EVA problem. As stated in Chapter 10, some of the well-known theoretical formulas, such as the Ackermann formula, the Bass-Gura formula, etc., though important in their own rights, do not yield to computationally viable methods. The primary reason is that these methods are based on transformation of the controllable pair (A, B) to the controller-companion form, and the transforming matrix can be highly ill-conditioned in some cases. The computationally viable methods for EVA should be based on transformation of the

pair (A, B) to the controller-Hessenberg form or the matrix A to the real Schur form, which can be achieved using orthogonal transformations. Several methods of this type have been developed in recent years and we have described a few of them in this chapter. The methods described here include

- A single-input recursive algorithm (**Algorithm 11.2.1**) (Datta (1987)) and its RQ implementation (**Algorithm 11.2.2**) (Arnold and Datta (1998))
- A multi-input generalization (**Algorithm 11.3.1**) of the single-input recursive algorithm (Arnold and Datta (1990)).
- A multi-input explicit QR algorithm (**Section 11.3.2**) (Miminis and Paige (1988)).
- A multi-input Schur algorithm (**Algorithm 11.3.3**) (Varga (1981)).

Algorithm 11.2.1 and Algorithm 11.3.1 are the fastest algorithms, respectively, for the single-input and the multi-input EVA problems.

Unfortunately, the numerical stability of these algorithms are not guaranteed. However, many numerical experiments performed by the author and others (e.g., Varga (1996), Arnold and Datta (1998)) show that Algorithm 11.2.1 works extremely well in practice, even with ill-conditioned problems. Furthermore, there is an RQ implementation which is numerically stable (Arnold and Datta (1998)). This stable version is described in **Algorithm 11.2.3**.

The **multi-input explicit QR** algorithm in **Section 11.3.2** is also numerically stable. **However, it might give a complex feedback in some cases.**

The **Schur algorithm** (**Algorithm 11.3.3**), based on the real-Schur decomposition of the matrix A , is most expensive; but, it has a distinguished feature that it can be used as a **partial-pole placement** algorithm in the sense that it lets the user reassign only a part of the spectrum leaving the rest unchanged. The algorithm is also believed to be **numerically stable**.

Besides the above mentioned algorithms, an algorithm (**Algorithm 11.6.1**) for **robust eigenvalue assignment**, which not only assigns a desired set of eigenvalues but also a set of well-conditioned eigenvectors as well, is also included in this Chapter. The robust eigenvalue assignment is important; because, the conditioning of the eigenvector matrix greatly influences the sensitivity of the closed-loop eigenvalues (see **Section 11.5**). Algorithm 11.6.1 is due to Kautsky, Nichols, and Van Dooren (1985) and is popularly known as the **KNV algorithm**. The MATLAB function **place** uses this algorithm.

Sections 11.4 and 11.5 are devoted, respectively, to the **conditioning of the feedback matrix and of the closed-loop eigenvalues**. The conditioning of the feedback matrix and the conditioning of the closed-loop eigenvalues are two different matters. It is easy to construct examples for which the feedback matrix can be computed rather very accurately by using a backward stable algorithm, but the resulting closed-loop eigenvalues might still be different from those to be assigned. These two problems are, therefore, treated separately.

The chapter concludes with a Table of **comparison of different methods** discussed in this chapter and **recommendations** are made based on this comparison.

11.2 Numerical Methods for the Single-input Eigenvalue Assignment Problem

The constructive proof of Theorem 10.4.1 suggests the following method for finding the feedback vector f .

Eigenvalue Assignment via Controller-Companion Form

Step 1. Find the coefficients d_1, d_2, \dots, d_n of the characteristic polynomial of the desired closed-loop matrix from the given set S .

Step 2. Transform (A, b) to the controller-companion form (C, \tilde{b}) :

$$TAT^{-1} = C, \quad Tb = \tilde{b},$$

where C is a **lower companion matrix** and $\tilde{b} = (0, 0, \dots, 0, 1)^T$.

Step 3. Compute $\hat{f}_i = d_i - a_i$, $i = 1, 2, \dots, n$ where a_i , $i = 1, \dots, n$ are the entries of the last row of C .

Step 4. Find $f^T = \hat{f}^T T$, where $\hat{f}^T = (\hat{f}_1, \hat{f}_2, \dots, \hat{f}_n)$.

Because of the difficulty of the implementation of Step 1 for large problems and of the instability of the algorithm due to possible ill-condition of T for finding the controllability canonical form in Step 2, as discussed in Chapter 6, **the above method is clearly not practical**.

Similar remarks hold for Ackermann's formula. The Ackermann (Ackermann (1972)) formula, though important in its own right, is not numerically effective. Recall that the Ackermann formula for computing f to assign the spectrum $S = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$ is:

$$f^T = e_n^T C_M^{-1} \phi(A),$$

where $C_M = (b, Ab, \dots, A^{n-1}b)$ and

$$\phi(x) = (x - \lambda_1)(x - \lambda_2) \cdots (x - \lambda_n).$$

Thus, the implementation of Ackermann's formula requires (i) computing $\phi(A)$ which involves computing various powers of A (ii) computing the last row of the inverse of the controllability matrix.

The **controllability matrix is usually ill-conditioned** (see the relevant comments in Chapter 6). The following example illustrates the point.

Example 11.2.1 Consider EVA using Ackermann's formula with

$$A = \begin{pmatrix} -4.0190 & 5.1200 & 0 & 0 & -2.0820 & 0 & 0 & 0 & 0.8700 \\ -0.3460 & 0.9860 & 0 & 0 & -2.3400 & 0 & 0 & 0 & 0.9700 \\ -7.9090 & 15.4070 & -4.0690 & 0 & -6.4500 & 0 & 0 & 0 & 2.6800 \\ -21.8160 & 35.6060 & -0.3390 & -3.8700 & -17.8000 & 0 & 0 & 0 & 7.3900 \\ -60.1960 & 98.1880 & -7.9070 & 0.3400 & -53.0080 & 0 & 0 & 0 & 20.4000 \\ 0 & 0 & 0 & 0 & 94.0000 & -147.2000 & 0 & 53.2000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 94.0000 & -147.20000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 12.80000 & 0 & -31.6000 & 0 \\ 0 & 0 & 0 & 0 & 12.8000 & 0 & 0 & 18.8000 & -31.6000 \end{pmatrix}$$

$$B = \begin{pmatrix} -0.1510 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad S = \begin{pmatrix} -1.0000 \\ -1.5000 \\ -2.0000 \\ -2.5000 \\ -3.0000 \\ -3.5000 \\ -4.0000 \\ -4.5000 \\ -5.0000 \end{pmatrix}$$

The eigenvalues assigned by the **Ackermann formula** are:

$$\begin{pmatrix} -0.8824 & - & 0.4891j \\ -0.8824 & + & 0.4891j \\ -2.2850 & - & 1.0806j \\ -2.2850 & + & 1.0806j \\ -3.0575 \\ -3.8532 \\ -4.2637 & - & 0.7289j \\ -4.2637 & + & 0.7289j \end{pmatrix}$$

Thus, the desired eigenvalues in S are completely different from those assigned by the Ackermann formula.

The same problem is then solved using the MATLAB function `place`, which uses the KNV algorithm.

The eigenvalues assigned by MATLAB function **place** are:

$$\begin{pmatrix} -4.9999 \\ -4.5001 \\ -4.0006 \\ -3.4999 \\ -3.0003 \\ -2.4984 \\ -2.0007 \\ -1.5004 \\ -0.9998 \end{pmatrix}$$

Similar results were obtained by the recursive Algorithm (**Algorithm 11.2.1**) (see Example 11.2.3).

A Template of Numerical Algorithms for EVA Problem

A practical numerically effective algorithm has to be based upon the reduction of the pair (A, B) to a controllable pair that uses a well-conditioned transformation. As we have seen in Chapter 6, the controller-Hessenberg pair is such a pair.

Indeed, several numerically effective algorithms have been proposed both for the single-input and the multi-input problems in recent years, based on the reduction of (A, B) to a controller-Hessenberg pair. We will describe some such algorithms in the sequel.

Most of these algorithms have a common basic structure which can be described as follows.

Step 1. The controllable pair (A, B) is first transformed to a controller-Hessenberg pair (H, \tilde{B}) ; that is, an orthogonal matrix P is constructed such that

$$PAP^T = H, \text{ an unreduced block upper Hessenberg matrix,}$$

$$PB = \tilde{B} = \begin{pmatrix} B_1 \\ 0 \end{pmatrix}, \text{ where } B_1 \text{ is upper triangular.}$$

Note: In the single-input case, the controller Hessenberg pair is (H, \bar{b}) , where H is an unreduced upper Hessenberg matrix and $\bar{b} = Pb = \beta e_1, \beta \neq 0$.

Step 2. The eigenvalue assignment problem is now solved for the pair (H, \tilde{B}) , by exploiting the special forms of H and \tilde{B} . This step involves finding a matrix F such that

$$\Omega(H - \tilde{B}F) = \{\lambda_1, \dots, \lambda_n\}.$$

(Here $\Omega(M)$ denotes the spectrum of the matrix M)

Note: In the single-input case, this step amounts to finding a row vector f^T such that $\Omega(H - \beta e_1 f^T) = \{\lambda_1, \dots, \lambda_n\}$.

Step 3. A feedback matrix K of the original problem is retrieved from the feedback matrix F of the transformed Hessenberg problem by using an orthogonal matrix multiplication:

$$K = FP.$$

Note that

$$\begin{aligned}\Omega(H - \tilde{B}F) &= \Omega(PAP^T - PBFPP^T) \\ &= \Omega(P(A - BK)P^T) \\ &= \Omega(A - BK).\end{aligned}$$

■

The different algorithms differ in the way Step 2 is implemented.

In describing the algorithms below, we will assume that Step 1 has already been implemented using the numerically stable stair-case algorithm described in Chapter 6 (**Section 6.8**).

11.2.1 A Recursive Algorithm for the Single-input EVA Problem

In this subsection, we present a simple recursive scheme for the single-input Hessenberg EVA problem. The scheme is due to Datta (1987).

Let's first remind the readers of the statement of the **single-input Hessenberg eigenvalue assignment problem**:

Given an unreduced upper Hessenberg matrix H , the number $\beta \neq 0$, and the set $S = \{\lambda_1, \dots, \lambda_n\}$, closed under complex conjugation, find a row vector f^T such that

$$\Omega(H - \beta e_1 f^T) = \{\lambda_1, \dots, \lambda_n\}.$$

We will assume temporarily, without any loss of generality, that $\beta = 1$ (recall that $Pb = \bar{b} = \beta e_1$.)

Formulation of the Algorithm

The single input eigenvalue assignment problem will have a solution if the closed-loop matrix $(H - e_1 f^T)$ can be made similar to a matrix whose eigenvalues are the same as the ones to be assigned.

Thus, the basic idea here is to construct a nonsingular matrix X such that

$$X(H - e_1 f^T)X^{-1} = \Lambda, \tag{11.2.1}$$

where $\Omega(\Lambda) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$.

From (11.2.1), we have

$$XH - \Lambda X = Xe_1 f^T. \quad (11.2.2)$$

Taking the transpose, the above equation becomes

$$H^T X^T - X^T \Lambda^T = f e_1^T X^T. \quad (11.2.3)$$

Setting $X^T = L$, the equation (11.2.3) becomes

$$H^T L - L \Lambda^T = f e_1^T L. \quad (11.2.4)$$

The problem at hand now is to construct a nonsingular matrix L and a vector f such that the equation (11.2.4) is satisfied. We show below how some special choices make it happen.

Let's choose

$$\Lambda^T = \begin{pmatrix} \lambda_1 & & & \\ * & \ddots & & 0 \\ & \ddots & \ddots & \\ 0 & \ddots & \ddots & \\ & & * & \lambda_n \end{pmatrix}$$

and let $e_1^T L$ be chosen such that all but the last column of the matrix on the right hand side of (11.2.4) are zero; that is, the matrix equation (11.2.4) becomes

$$H^T L - L \Lambda^T = (0, 0, \dots, \alpha f), \alpha \neq 0. \quad (11.2.5)$$

The simple form of the right hand side of (11.2.5) allows us compute recursively the second through n th column of $L = (l_1, l_2, \dots, l_n)$, once the first column l_1 is known. The entries of the subdiagonal of Λ can be chosen as scaling factors for the computed columns of L . Once L is known, αf can be computed by equating the last column of both sides of (11.2.5):

$$\alpha f = (H^T - \lambda_n I) l_n \quad (11.2.6)$$

What now remains to be shown is that how to choose l_1 such that the resulting matrix L in (11.2.5) is nonsingular.

A theorem of K. Datta (1988) tells us that if l_1 is chosen such that (H^T, l_1) is controllable, then L satisfying (11.2.5) will be nonsingular. Since H^T is an unreduced upper Hessenberg matrix, the simplest choice is $l_1 = e_n = (0, 0, \dots, 0, 1)^T$. It is easy to see that this choice of l_1 will yield $\alpha = l_{1n}$, the first entry of l_n . Then equating the last column of both sides of (11.2.5), we have

$$f = \frac{(H^T - \lambda_n I) l_n}{\alpha} = \frac{(H^T - \lambda_n I) l_n}{l_{1n}}.$$

The above discussion immediately leads us to the following algorithm:

Algorithm 11.2.1 The Recursive Algorithm for the Single-Input Hessenberg EVA Problem

Inputs: H , an unreduced upper Hessenberg matrix, and $S = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$, a set of n numbers, closed under complex conjugation

Output: The feedback vector f such that $\Omega(H - e_1 f^T) = S$.

Step 1. Set $l_1 = e_n$, the last column of the $n \times n$ identity matrix

Step 2. Construct a set of normalized vectors $\{\ell_k\}$ as follows:

For $i = 1, 2, \dots, n-1$ do

Compute $\hat{\ell}_{i+1} = (H^T - \lambda_i I)\ell_i$

$$\ell_{i+1} = \frac{\hat{\ell}_{i+1}}{\|\hat{\ell}_{i+1}\|_2}$$

End

Step 3. Compute $\ell_{n+1} = (H^T - \lambda_n I)\ell_n$.

Step 4. Compute $f = \frac{\ell_{n+1}}{\alpha}$, where α is the first entry of ℓ_n .

Theorem 11.2.1 The vector f computed by Algorithm 11.2.1 is such that the eigenvalues of the closed-loop matrix $(H - e_1 f^T)$ are $\lambda_1, \dots, \lambda_n$.

Proof: Proof follows from the above discussions. ■

Flop-Count. Since l_i contains only i nonzero entries and H is an unreduced Hessenberg matrix, computations of l_2 through l_n in the Algorithm 11.2.1 takes about $\frac{n^3}{3}$ flops. Furthermore, with these operations, one gets the transforming matrix L that transforms the closed loop matrix to Λ by similarity. Also, it takes about $\frac{10}{3}n^3$ flops for controller-Hessenberg reduction. So, the flop-count for the EVA problem for the pair (A, b) using Algorithm 11.2.1 is $\frac{11}{3}n^3$ flops.

Avoiding Complex Arithmetic. When the eigenvalues to be assigned are complex, the use of complex arithmetic in Algorithm 11.2.1 can be avoided by setting Λ as a tridiagonal matrix in real Schur form, having a 2×2 block corresponding to each pair of complex conjugate eigenvalues to be assigned. Algorithm 11.2.1 needs to be modified accordingly [Exercise 1].

MATCONTROL NOTE: The modified algorithm that avoids the use of complex arithmetic has been implemented in MATCONTROL function **polercs**.

Example 11.2.2 Consider EVA using Algorithm 11.2.1 with

$$H = \begin{pmatrix} 9 & 4 & 7 \\ 3 & 1 & 2 \\ 0 & 9 & 6 \end{pmatrix}, \quad S = \{9 \ 5 \ 1\}.$$

Step 1. $l_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

Step 2. $i = 1$:

$$\hat{l}_2 = \begin{pmatrix} 0 \\ 9 \\ -3 \end{pmatrix}, \quad l_2 = \begin{pmatrix} 0 \\ 0.9487 \\ -0.3162 \end{pmatrix}.$$

$i = 2$:

$$\hat{l}_3 = \begin{pmatrix} 2.8460 \\ -6.6408 \\ 1.5811 \end{pmatrix}, \quad l_3 = \begin{pmatrix} 0.3848 \\ -0.8979 \\ 0.2138 \end{pmatrix}$$

Step 3.

$$l_4 = \begin{pmatrix} 0.3848 \\ 3.4633 \\ 1.9668 \end{pmatrix}.$$

Step 4.

$$f = \begin{pmatrix} 1.0000 \\ 9.0000 \\ 5.1111 \end{pmatrix}$$

The closed-loop matrix:

$$H - e_1 f^T = \begin{pmatrix} 8.0 & -5.0 & 1.8889 \\ 3.0 & 1.0 & 2.0 \\ 0 & 9.0 & 6.0 \end{pmatrix}$$

Verify: The eigenvalues of the matrix $(H - e_1 f^T)$ are 9, 5, and 1.

Example 11.2.3

Let's apply Algorithm 11.2.1 to Example 11.2.1 again. The eigenvalues of the closed-loop matrix $H - e_1 f^T$ with the vector f computed by Algorithm 11.2.1 are:

$$-5.0001, -4.4996, -4.0009, -3.4981, -3.0034, -2.4958, -2.0031, -1.4988, 1.0002.$$

The computed closed-loop eigenvalues thus, have the same accuracy as those obtained by the MATLAB function place.

Example 11.2.4

Since the matrix H and the closed-loop matrix $H - e_1 f^T$ differ only by the first row, Algorithm 11.2.1 amounts to finding a vector f such that, when the first row of the given Hessenberg matrix H is replaced by f^T , the resulting new Hessenberg matrix has the prescribed spectrum. Let H be the well-known Wilkinson bidiagonal matrix (see Datta (1995), Wilkinson (1965)) with highly ill-conditioned eigenvalues:

$$H = \begin{pmatrix} 20 & & & \\ & 19 & & \\ 20 & \ddots & \ddots & 0 \\ & 20 & \ddots & \\ 0 & \ddots & 20 & \ddots \\ & & & 1 \end{pmatrix}$$

First, the first row of H is replaced by the zero row vector and then Algorithm 11.2.1 is run on this new H with $S = \{20, 19, 18, \dots, 1\}$, and f is computed. Since the eigenvalues of the original matrix H is the set S ; in theory, f^T should be the same as the first row of H ; namely, $f^T = (20, 0, \dots, 0)$. Indeed, the vector f^T computed by the algorithm is found to be $f^T = (20, 0, \dots, 0)$ and the eigenvalues of the closed-loop matrix with this computed f are $1, 2, 3, \dots, 20$.

A closed-form solution of the single-input eigenvalue assignment problem.

We now show that Algorithm 11.2.1 yields an explicit closed-form solution for the single-input problem. The recursion in Step 2 of the algorithm yields

$$\gamma l_{i+1} = (H^T - \lambda_1 I)(H^T - \lambda_2 I) \cdots (H^T - \lambda_i I)l_1, \quad (11.2.7)$$

for some (nonzero) scalar γ . Including Steps 1 and 4, (11.2.6) becomes

$$\alpha f = (H^T - \lambda_1 I)(H^T - \lambda_2 I) \cdots (H^T - \lambda_n I)e_n, \quad (11.2.8)$$

where $\alpha = (h_{21}h_{32} \cdots h_{n,n-1})^{-1}$. If $\phi(x) = (x - \lambda_1)(x - \lambda_2) \cdots (x - \lambda_n)$, then this can be written as

$$f^T = \alpha e_n^T \phi(H). \quad (11.2.9)$$

Since this solution is unique, it represents **the Hessenberg representation of the Ackermann formula for the single-input eigenvalue assignment problem**.

11.2.2 An Error Analysis of the Recursive Single-Input Method

Knowing the above explicit expression for the feedback vector f , we can now carry out a forward error analysis of Algorithm 11.2.1. This is presented below.

By duality of (11.2.4), we see that the method computes a matrix L and a vector f such that

$$HL - L\Lambda = \alpha f e_1^T L.$$

A careful look at the iteration reveals that the forward error has a special form. Define the polynomials $\phi_{j,k}$ for $j \leq k$ by

$$\phi_{j,k}(x) = (x - \lambda_j)(x - \lambda_{j+1}) \cdots (x - \lambda_k).$$

Let \bar{l}_i be the computed value of the i th column of L . Define ϵ_i by

$$\bar{l}_{i+1} = (H - \lambda_i I)\bar{l}_i + \epsilon_i. \quad (11.2.10)$$

Then we have the following **forward error** formula, due to Arnold (1993) (See also Arnold and Datta (1998)).

Theorem 11.2.2 *Let $\bar{\alpha}\bar{f}$ be the computed feedback vector of the single-input eigenvalue assignment problem for (H, e_1) using Algorithm 11.2.1. If αf is the exact solution, then*

$$\bar{\alpha}\bar{f} - \alpha f = \sum_{j=1}^n \phi_{j,n}(H)\epsilon_j,$$

where ϵ_j 's are defined above.

Unfortunately, not much can be said about backward stability from a result like this.

It is, however, possible to shed some light on the stability of this method by looking at ϵ_j in a different way.

Theorem 11.2.3 *Let $E = [\epsilon_1, \epsilon_2, \dots, \epsilon_n]$ and let $\bar{L} = [\bar{l}_1, \bar{l}_2, \dots, \bar{l}_n]$. Then $\bar{\alpha}\bar{f}$ solves (exactly) the single-input EAP for the perturbed system $(H - E\bar{L}^{-1}, \beta e_1, S)$, where the ϵ_i are the same as in the previous theorem; that is, the computed vector \bar{f} is such that*

$$\Omega[(H - E\bar{L}^{-1}) - \beta e_1 \bar{f}^T] = \{\lambda_1, \dots, \lambda_n\}.$$

Proof: Notice that as defined, \bar{L} satisfies the matrix equation

$$H\bar{L} - \bar{L}\Lambda = E + \bar{\alpha}\bar{f}e_n^T,$$

where $\Lambda = \text{diag}(\lambda_i)$. Since \bar{L} is nonsingular by construction, we can solve the perturbed equation

$$(H + \Delta H)\bar{L} - \bar{L}\Lambda = E + \bar{\alpha}\bar{f}e_n^T$$

for ΔH . This yields $-\Delta H = E\bar{L}^{-1}$, and $\bar{\alpha}\bar{f}$ solves the EAP for $(H + \Delta H, \beta e_1, S)$.

■

Remarks on Stability and Reliability

From the above result it can not be said that the method is backward stable. The result simply provides an upper bound on the size of the ball around the initial data, inside which there exist $(H + \Delta H, \beta + \delta\beta)$ for which the computed solution is exact. If $\|\Delta H\|$ could be bounded above by a small quantity that was relatively independent of the initial data, then the method would be backward stable. However, Theorem 11.2.3 does allow one to say precisely when the results from the method are suspect. It is clear that $\|E\|$ is always small if the iterates are normalized every few steps, so that all of the backward error information is contained in \bar{L}^{-1} . **Thus, the stability of the algorithm can be monitored or controlled by monitoring the condition number of \bar{L} . Furthermore, since \bar{L} is triangular, it is possible to estimate $\|\bar{L}^{-1}\|$ rather cheaply, even as the iteration proceeds.** (See Higham (1996)). The matrix L gives us even more information about the eigenvalue assignment problem at hand. In case the eigenvalues to be assigned are distinct, an upper bound in the condition number of the matrix of eigenvectors that diagonalizes the closed loop matrix can be obtained from the condition number of the matrix L .

This is important, because, as said in the introduction, the condition number of the matrix of eigenvectors of the closed-loop matrix is an important factor in the accuracy of the assigned eigenvalues (see Section 11.5 and the Example therein).

Specifically, if X diagonalizes Λ , then it can be shown (Arnold and Datta (1998)), that $P = (\bar{L})^{-1}X$ diagonalizes the closed-loop matrix $H - e_1 f^T$, furthermore,

$$\text{Cond}_2(P) \leq \text{Cond}_2(X)\text{Cond}_2(L).$$

Computational experience has shown that if L is ill-conditioned, then so are the closed-loop eigenvalues.

11.2.3 The QR and RQ Implementations of Algorithm 11.2.1

Algorithm 11.2.1 is an extremely efficient way to solve the Hessenberg single-input EVA problem, but as we have just seen, the backward stability of this algorithm cannot be guaranteed. It, however, turns out that there is a **numerically stable implementation** of this algorithm via QR iterations. We will discuss this below.

The QR Implementation of Algorithm 11.2.1

The idea of using QR iterations in implementing Algorithm 11.2.1 comes from the fact that the matrix $\phi(H)$ in the explicit expression of f in (11.2.9) can be written as (Stewart (1973), 353):

$$\phi(H) = (H - \lambda_2 I)(H - \lambda_2 I) \cdots (H - \lambda_n I) = Q_1 Q_2 \cdots Q_n R_n R_{n-1} \cdots R_1,$$

where Q_i and R_i are generated by QR iterations as follows:

$H_1 = H$

For $i = 1, 2, \dots, n$ do

$$Q_i R_i = H_i - \lambda_i I$$

$$H_{i+1} = R_i Q_i + \lambda_i I$$

End.

MATCONTROL NOTE: The QR version of Algorithm 11.2.1 has been implemented in MATCONTROL function **poleqrs**.

The RQ Implementation of Algorithm 11.2.1

The difficulty of implementing the QR strategy is that the R_i need to be accumulated; the process is both expensive and unstable.

We now show how the method can be made computationally efficient by using RQ factorizations instead of the QR factorizations, as follows:

Set $H_1 = H$

For $i = 1, 2, \dots, n$ compute the RQ step

$$R_i Q_i = H_i - \lambda_i I$$

$$H_{i+1} = Q_i R_i + \lambda_i I$$

This time

$$\phi(H) = R_1 R_2 \cdots R_n Q_n Q_{n-1} \cdots Q_1, \quad (11.2.11)$$

and by setting $Q = Q_n Q_{n-1} \cdots Q_1$ and $R = R_1 R_2 \cdots R_n$, we have from (11.2.9)

$$f^T = \alpha e_n^T R Q = \alpha \rho e_n^T Q, \quad (11.2.12)$$

where $\rho = \prod_{i=1}^n r_{nn}^{(i)}$ where $r_{nn}^{(i)}$ denotes the (n, n) th entry of R_i . This is a much nicer situation! Thus a straightforward RQ implementation of Algorithm 11.2.1 will be as follows:

Algorithm 11.2.2 (An RQ Implementation of Algorithm 11.2.1)

Inputs: Same as in Algorithm 11.2.1

Output: Same as in Algorithm 11.2.1

Step 0. Set $H_1 = H$.

Step 1. For $i = 1, 2, \dots, n$ do

$$R_i Q_i = H_i - \lambda_i I$$

$$H_{i+1} = Q_i R_i + \lambda_i I$$

End

Step 2. Compute $f = \alpha \rho e_n^T Q_n Q_{n-1} \cdots Q_1$, where $\rho = \prod_{i=1}^n r_{nn}^{(i)}$, $r_{nn}^{(i)}$ denotes the (n, n) th entry of R_i .



Example 11.2.5 Consider Example 11.2.2 again.

Step 0. $H_1 = \begin{pmatrix} 9 & 4 & 7 \\ 3 & 1 & 2 \\ 0 & 9 & 6 \end{pmatrix}$, $S = \{\lambda_1, \lambda_2, \lambda_3\} = \{9, 5, 1\}$.

Step 1. $i = 1$. $[R_1, Q_1] = \mathbf{rq}(H_1 - \lambda_1 I)$ gives

$$Q_1 = \begin{pmatrix} -0.2063 & -0.3094 & -0.92833 \\ 0.9785 & -0.0652 & -0.1957 \\ 0 & 0.99487 & -0.3162 \end{pmatrix}$$

$$R_1 = \begin{pmatrix} -7.7357 & -1.6308 & 1.5811 \\ 0 & 3.0659 & -8.2219 \\ 0 & 0 & 9.4868 \end{pmatrix}$$

$$H_2 = Q_1 R_1 + \lambda_1 I = \begin{pmatrix} 10.5957 & -0.6123 & -6.5885 \\ -7.5693 & 7.2043 & 0.2269 \\ 0 & 2.9086 & -1.8000 \end{pmatrix}$$

$i = 2$

$[R_2, Q_2] = \mathbf{rq}(H_2 - \lambda_2 I)$ give

$$R_2 = \begin{pmatrix} 1.5311 & 6.2383 & 5.8168 \\ 0 & -7.8595 & 0.6582 \\ 0 & 0 & 7.3959 \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} -0.2692 & -0.8855 & -0.3788 \\ 0.9631 & -0.2475 & -0.1059 \\ 0 & 0.3933 & -0.9194 \end{pmatrix}$$

$$H_3 = Q_2 R_2 + \lambda_2 I = \begin{pmatrix} 4.5878 & 5.2799 & -4.9501 \\ 1.4746 & 12.9533 & 4.6561 \\ 0 & -3.0909 & -1.5411 \end{pmatrix}$$

$i = 3$

$[R_3, Q_3] = \mathbf{rq}(H_3 - \lambda_3 I)$

gives

$$R_3 = \begin{pmatrix} -0.8804 & -7.9752 & -0.9349 \\ 0 & -4.2580 & -12.1904 \\ 0 & 0 & 4.0014 \end{pmatrix}$$

$$Q_3 = \begin{pmatrix} -0.9381 & 0.2199 & -0.2675 \\ -0.3463 & -0.5958 & 0.7247 \\ 0 & -0.7725 & -0.6351 \end{pmatrix}$$

Step 2. $\alpha = \frac{1}{27}$, $\rho = r_{33}^{(1)} r_{33}^{(2)} r_{33}^{(3)} = 280.7526$, $f^T = \alpha \rho e_n^T Q_3 Q_2 Q_1 = (1, 9, 5.1111)$.

Verify: $\Omega(H - e_1 f^T) = \{5, 1, 9\}$.

As Example 11.2.5 shows Algorithm 11.2.2 may be made storage-efficient by observing that it is possible to **deflate** the problem at each RQ step, as follows:

$$H_{i+1} = Q_i R_i + \lambda_i I = \begin{pmatrix} * & * \\ 0 & \tilde{H}_{i+1} \end{pmatrix}.$$

The matrix \tilde{H}_{i+1} now can be set as H_{i+1} and the iteration can be continued with $H_{i+1} \equiv \tilde{H}_{i+1}$ after updating Q_i and ρ appropriately. Thus, algorithmically we have the **following storage-efficient version of Algorithm 11.2.2, which is recommended to be used in practice.**

Algorithm 11.2.3 - A Storage Efficient version of Algorithm 11.2.2

Inputs: Same as in Algorithm 11.2.1

Output: Same as in Algorithm 11.2.1

Step 0. Set $H_1 = H$.

Step 1. Compute the RQ factorization of $H_1 - \lambda_1 I$;

that is, compute Q_1 and R_1 such that $(H - \lambda_1 I)Q_1^T = R_1$.

Compute $H_2 = Q_1 R_1 + \lambda_1 I$.

Set $Q = Q_1$ and $\rho = r_{nn}^{(1)}$, where $R_1 = (r_{ij}^{(1)})$.

Step 2.

For $i = 2, 3, \dots, n$ do

Compute the RQ factorization of $H_i - \lambda_i I$: $(H_i - \lambda_i I)Q_i^T = R_i$.

Compute H_{i+1} , where $Q_i R_i + \lambda_i I = \begin{pmatrix} * & * \\ 0 & H_{i+1} \end{pmatrix}$.

Update $Q \equiv \begin{pmatrix} I & \\ & Q_i \end{pmatrix} Q$, where I is a matrix consisting of

the first $(i-2)$ rows and columns of the identity matrix.

Update $\rho \equiv \rho r_{n+2-i,n+2-i}^{(i)}$ ($r_{n+2-i,n+2-i}^{(i)}$ is the last element of R_i)

End

Step 3. Compute $f^T = \alpha' \rho e_n^T Q$, where $\alpha' = \frac{1}{h_{21} \dots h_{n,n-1}}$.

Flop-Count and Numerical Stability

Algorithm 11.2.3 requires about $\frac{5}{3}n^3$ flops. Since reduction to the controller-Hessenberg form requires $\frac{10}{3}n^3$ flops, the total count for EVA of the pair (A, b) using Algorithm 11.2.3 is about $5n^3$.

The algorithm is *numerically stable* (see Arnold and Datta (1998)). Specifically, the method computes, given a controllable pair (H, e_1) , a vector \bar{f} such that it solves exactly the eigenvalue assignment problem for the system with the matrix $H + \Delta H$ where

$$\|\Delta H\|_F \leq \mu g(n) \|H\|_F,$$

μ is the machine precision, $g(n)$ is a modest function of n .

Remarks: It can be shown (Arnold (1993)) that if an eigenvalue assignment algorithm for the single-input Hessenberg problem is backward stable, then the algorithm is also backward stable for the original problem.

Thus the RQ implementation of Algorithm 11.2.1 is backward stable for the original problem. That is, the feedback \bar{k} , computed by Algorithm 11.2.3, for the problem (A, b) is exact for a nearby problem: \bar{k} exactly solves the EAP for $(A + \Delta A, b + \delta b)$, where ΔA and δb are small. For a proof of this backward stability result, see Arnold and Datta (1998).

Example 11.2.6 Consider Example 11.2.2 again.

Step 0.

$$H_1 = H = \begin{pmatrix} 9 & 4 & 7 \\ 3 & 1 & 2 \\ 0 & 9 & 6 \end{pmatrix},$$

$$S = \{\lambda_1, \lambda_2, \lambda_3\} = \{9, 5, 1\}.$$

Step 1.

Compute R_1 and Q_1 such that $H_1 - \lambda_1 I = R_1 Q_1 : [R_1, Q_1] = \mathbf{r} \mathbf{q}$ ($H_1 - \lambda_1 I$).

R_1 and Q_1 are the same as in Example 11.2.5.

$$\text{Compute } H_2 = Q_1 R_1 + \lambda_1 I = \begin{pmatrix} 10.5957 & -0.6123 & -6.5885 \\ -7.5693 & 7.2043 & 0.2269 \\ 0 & 2.9086 & -1.8000 \end{pmatrix}$$

$$Q \equiv Q_1 = \begin{pmatrix} -0.2063 & 0.3094 & 0.9283 \\ 0.9785 & -0.0652 & -0.1957 \\ 0 & 0.9487 & -0.3162 \end{pmatrix}$$

$$\rho = 9.4868$$

Step 2.

$i = 2$

Compute R_2 and Q_2 such that $H_2 - \lambda_2 I = R_2 Q_2 : [R_2, Q_2] = \mathbf{r} \mathbf{q}$ ($H_2 - \lambda_2 I$)

R_2 and Q_2 are the same as in Example 11.2.5.

$$H_3 = \begin{pmatrix} 12.9533 & 4.6561 \\ -3.0909 & -1.5411 \end{pmatrix}.$$

$$\text{Update } Q : Q \equiv \begin{pmatrix} -0.8109 & -0.2183 & 0.5430 \\ -0.4409 & -0.3823 & -0.8121 \\ 0.3848 & -0.8479 & 0.2138 \end{pmatrix}.$$

$$\text{Update } \rho : \rho \equiv 70.1641$$

$i = 3$

Compute R_3 and Q_3 such that $H_3 - \lambda_3 I = R_3 Q_3$:

$$R_3 = \begin{pmatrix} -3.9945 & -12.1904 \\ 0 & 4.0014 \end{pmatrix},$$

$$Q_3 = \begin{pmatrix} -0.6351 & 0.7725 \\ -0.7725 & -0.6351 \end{pmatrix}.$$

$$H_4 = 7.8755$$

$$\text{Update } Q : Q \equiv \begin{pmatrix} -0.8109 & -0.2183 & 0.5430 \\ 0.5772 & -0.4508 & 0.6809 \\ 0.0962 & 0.8655 & 0.4915 \end{pmatrix}.$$

$$\text{Update } \rho \equiv 280.7526.$$

Step 3.

$$f = \begin{pmatrix} 1.0000 \\ 9.0000 \\ 5.1111 \end{pmatrix}.$$

Verify:

$$H - e_1 f^T = \begin{pmatrix} 8.0000 & -5.0000 & 1.8889 \\ 3.0000 & 1.0000 & 2.0000 \\ 0 & 9.0000 & 6.0000 \end{pmatrix}.$$

The eigenvalues of $H - e_1 f^T$ are $\{5, 1, 9\}$.

MATCONTROL NOTE: Algorithm 11.2.3 has been implemented in MATCONTROL function **polerqs**.

11.2.4 Explicit and Implicit RQ Algorithms

We have just seen the QR and RQ versions of algorithm 11.2.1. At least two more QR type methods were proposed earlier: An **explicit QR algorithm** by Miminis and Paige (1982) and an **implicit QR** algorithm by Patel and Misra (1984).

The explicit QR algorithm, proposed by Miminis and Paige (1982), constructs an orthogonal matrix Q such that

$$Q^T(H - \beta e_1 f^T)Q = R,$$

where R is an upper triangular matrix with the desired eigenvalues $\lambda_1, \dots, \lambda_n$ on the diagonal. The algorithm has a forward sweep and a backward sweep. The forward sweep determines Q and the backward sweep finds f and R , if needed. The algorithm explicitly uses the eigenvalues to be assigned as shifts and that is why it is called an **explicit QR algorithm**.

It should come as no surprise that **an implicit** RQ step is possible, and in order to handle **complex pairs of eigenvalues with real arithmetic**, an implicit double step is needed. One such method has been proposed by Patel and Misra (1984). The Patel-Misra method is similar to the Miminis-Paige method, but it includes an alternative to the “backward sweep”, There now exist RQ formulations of both these algorithms (Arnold (1993), Arnold and Datta (1998)). These RQ formulations are easier to describe, understand, and implement. We state the RQ versions of these two algorithms in **Exercises 9 and 10**, respectively.

It should be mentioned here that there now exists a generalization of the implicit QR algorithm due to Varga (1996) that performs an implicit multi-step in place of a double-step. The Varga algorithm is slightly more efficient than the Patel-Misra algorithm and like the latter, is believed to be numerically stable. We refer the readers to the paper of Varga (1996) for details.

Methods Not Discussed

Besides the methods discussed above, there are many other methods for the single-input problem. These include the methods based on solutions of independent linear systems (Datta and Datta (1986), Bru, Mas, and Urbano (1994a); the eigenvector method by Petkov, Christov, and Konstantinov (1984), etc., parallel algorithms by Coutinho et al. (1995), and by Bru et al. (1994c), etc.; and the multishift algorithm by Varga (1996). See, **Exercises 2, 3, 4 and 8** for statements of some of these methods.

11.3 Numerical Methods for the Multi-input Eigenvalue Assignment Problem

Some of the single-input algorithms described in the last section have been generalized in a straightforward fashion to the multi-input case or similar algorithms have been developed for the latter.

We describe here

- A multi-input generalization of the single-input recursive algorithm (Arnold and Datta (1990)).
- An explicit QR algorithm (Miminis and Paige (1988)).
- A Schur method (Varga (1981)).

There are many more algorithms for this problem that are not described here. Some of them are: a multi-input generalization of the single-input eigenvector algorithm by Petkov et al. (1986), a multi-input generalization of the single-input algorithm using solutions of linear systems by Bru et al. (1994b) [**Exercise 5**], a matrix equation algorithm by Bhattacharyya and DeSouza (1982) [**Exercise 14**], and a multi-input version of the single-input implicit QR algorithm by Patel and Misra (1984), algorithms by Tsui (1986) and Shafai and Bhattacharyya (1988) and, parallel algorithms by Baksi, Datta and Roy (1994), Datta (1989), etc.

11.3.1 A Recursive Multi-input Eigenvalue Assignment Algorithm

The following algorithm is a generalization of the single-input recursive algorithm (**Algorithm 11.2.1**) to the multi-input case.

The development of this algorithm is along the same line as Algorithm 11.2.1.

The version of the algorithm presented here is little different than that originally proposed in Arnold and Datta (1990).

Given a controller-Hessenberg pair (H, \tilde{B}) and the set $S = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$, the algorithm, like its single-input version, constructs a nonsingular matrix L recursively, from where the feedback matrix F can be easily computed. Since in the multi-input case the matrix H of the controller Hessenberg form is a block Hessenberg matrix, by taking advantage of the block form of H , the matrix L this time can be computed in blocks. The matrix L can be computed either block column-wise or block row-wise. We compute L block row-wise here starting with the last block row.

Thus, setting

$$\Lambda = \begin{pmatrix} \Lambda_{11} & & & \\ \Lambda_{21} & \Lambda_{22} & & 0 \\ \ddots & \ddots & & \\ 0 & & \Lambda_{k,k-1} & \Lambda_{kk} \end{pmatrix}$$

where the eigenvalues $\lambda_1, \dots, \lambda_n$ are contained in the diagonal blocks of Λ , and considering the equation:

$$LH - \Lambda L = L \begin{pmatrix} R \\ 0 \end{pmatrix} F,$$

it is easily seen that the matrices L and R can be found without knowing the matrix F . Indeed,

the matrix $L = \begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_k \end{pmatrix}$ can be computed recursively block row-wise starting with L_k and if L_k is chosen as $L_k = (0, 0, \dots, 0, I_{n_k})$, then L will be nonsingular. Equating the corresponding block-rows of the equation

$$LH - \Lambda H = \begin{pmatrix} R \\ 0 \end{pmatrix} F,$$

it is easy to see that

$$\Lambda_{i+1,i} L_i = L_{i+1} H - \Lambda_{i+1,i+1} L_{i+1} = \tilde{L}_i, i = k-1, k-2, \dots, 2, 1,$$

from where the matrices $\Lambda_{i+1,i}$ and L_i can be computed by using the QR factorization of \tilde{L}_i . Once L and R are found, the matrix F can be computed from the above equation by solving a block linear system. Overall, we have the following algorithm.

Algorithm 11.3.1 The Recursive Algorithm for the Multi-input EVA Problem

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix ($m \leq n$).

S - The set of numbers $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$, closed under complex conjugation.

Assumption : (A, B) is controllable.

Output: A feedback matrix K such that $\Omega(A - BK) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$.

Step 1. Using the stair-case algorithm in Section 6.7, reduce the pair (A, B) to the controller-Hessenberg pair (H, \tilde{B}) ; that is find an orthogonal matrix P such that $PAP^T = H$, an unreduced block upper Hessenberg matrix with k diagonal blocks

and

$$PB = \tilde{B} = \begin{pmatrix} R \\ 0 \end{pmatrix}, \text{ } R \text{ is upper triangular and has full rank.}$$

Step 2. Partition S in such a way that $S = \cup \Omega(\Lambda_{ii})$, where each Λ_{ii} is an $n_i \times n_i$ diagonal matrix (Recall that n_i 's are defined by the dimensions of the blocks in $H = (H_{ij})$; $H_{ij} \in \mathbb{R}^{n_i \times n_j}$).

Step 3. Set $L_k = (0, \dots, 0, I_{n_k})$.

Step 4. For $i = k-1, \dots, 1$ do

4.1 Compute $\tilde{L}_i \equiv L_{i+1} H - \Lambda_{i+1,i+1} L_{i+1}$

4.2 Compute the QR decomposition of $\tilde{L}_i^T : \tilde{L}_i^T = QR$

4.3 Define $L_i = Q_1^T$, where Q_1 are the first n_i columns of the matrix $Q = (Q_1, Q_2)$

End

Step 5. Solve the linear system $(L_{11}R)F = L_1H - \Lambda_{11}L_1$ for F , where L_{11} is the matrix of the first n_1 columns of L_1 .

Step 6. Compute the feedback matrix K of the original problem: $K \equiv FP$

Theorem 11.3.1 The feedback matrix K constructed by the above algorithm is such that

$$\Omega(A - BK) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}.$$

Proof: Proof follows from the discussion preceding the algorithm. ■

Flop-count: Approximately $\frac{19}{3}n^3 + \frac{15}{2}n^2m$ flops are required to implement the algorithm.

It may be worth noting a few points regarding the complexity of this algorithm. First, the given operations count includes assigning complex eigenvalues using real arithmetic. Second, almost 95% of the total flops required for this method are in the reduction to the controller-Hessenberg form in step 1. Finally, within the above operations count (but with some obvious additional storage requirements), the matrix L that transforms the reduced closed-loop system to the block bidiagonal matrix Δ by similarly, can be obtained.

Avoiding Complex Arithmetic: In order to assign a pair of complex conjugate eigenvalues using only real arithmetic, we set 2×2 “**Schur bumps**” on the otherwise diagonal Λ_{ii} . For example, if we want to assign the eigenvalues $x \pm iy$ to $A - BK$, we might set $\Lambda_3 = \begin{bmatrix} x & -y \\ y & x \end{bmatrix}$. However, the algorithm might give a complex feedback matrix if all the complex conjugate pairs cannot be distributed as above along the diagonal blocks Λ_{ii} . Some modifications of the algorithm in that case will be necessary. A *block algorithm that avoids complex feedback has been recently proposed by Carvalho and Datta (2001)*.

MATCONTROL NOTE: The modified version of Algorithm 11.3.1, proposed in Carvalho and Datta (2001), that avoids the use of complex arithmetic and is guaranteed to give a real feedback matrix has been implemented in MATCONTROL function **polercx**, while Algorithm 11.3.1 as it appears here has been implemented in MATCONTROL function **polercm**.

Example 11.3.1 Consider EVA using Algorithm 11.3.1 with

$$A = H = \begin{pmatrix} 1 & 2 & 3 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$B = \tilde{B} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad S = \{1, 2, 3, 4, 5\}.$$

$$\begin{aligned} k &= 3 \\ n_1 &= 3 \\ n_2 &= 1 \\ n_3 &= 1 \end{aligned}$$

Step 2. $\Lambda_{11} = \text{diag}(1, 2, 3)$, $\Lambda_{22} = 4$, $\Lambda_{33} = 5$.

Step 3. $L_3 = (0, 0, 0, 0, 1)$.

Step 4. $i = 2$

Step 4.1: $\tilde{L}_2 = (0 \ 0 \ 0 \ 1 \ -4)$

Step 4.3: $L_2 = (0 \ 0 \ 0 \ -0.2425 \ 0.9701)$

$i = 1$

Step 4.1: $\tilde{L}_1 = (0 \ 0 \ -0.2425 \ 1.6977 \ -3.3955)$

Step 4.3:

$$L_1 = \begin{pmatrix} 0 & 0 & 0.0638 & -0.4463 & 0.8926 \\ 0 & 1.0000 & 0 & 0 & 0 \\ 0.0638 & 0 & 0.9959 & 0.0285 & -0.0569 \end{pmatrix}$$

Step 5.

$$F = \begin{pmatrix} -2.3333 & 3.3333 & 78.2161 & -212.4740 & 217.0333 \\ -0.3333 & -1.6667 & 5.6667 & -9.0000 & 9.6667 \\ 0.6667 & 0.3333 & -2.3333 & 5.0000 & -4.3333 \end{pmatrix}$$

Verify: The eigenvalues of $H - \tilde{B}F$ are: {1.0000 2.0000 3.0000 4.0000 5.0000}.

11.3.2 The Explicit QR Algorithm for the Multi-input EVA Problem

The following multi-input QR algorithm due to Miminis and Paige (1988) also follows the same “template” as of the proceeding algorithm. The algorithm consists of the following three major steps.

Step 1. The controllable pair (A, B) is transformed to the controller-Hessenberg pair (H, \tilde{B}) :

$$PAP^T = H = \begin{pmatrix} H_{11} & & & & \\ H_{21} & \ddots & & H_{ij} & \\ \ddots & \ddots & \ddots & & \\ 0 & & H_{k,k-1} & H_{kk} & \end{pmatrix},$$

and $PBU = \tilde{B} = \begin{pmatrix} B_{11} \\ 0 \end{pmatrix}$. The matrix B_{11} and the subdiagonal blocks in H are of the form $(0, R)$, where R is a nonsingular and upper triangular matrix.

Step 2. An orthogonal matrix Q and a feedback matrix F are constructed such that $\Omega(Q^T(H - \tilde{B}F)Q) = \{\lambda_1, \dots, \lambda_n\}$.

Step 3. A feedback matrix K of the original problem is recovered from the feedback matrix F of the Hessenberg problem in Step 2 as follows:

$$K = UFP.$$

Step 1 can be implemented using the staircase algorithm for the controller-Hessenberg form described in Chapter 6.

We therefore concentrate on Step 2, assuming that Step 1 has already been performed.

Let n_1 = dimensions of H_{11} and $n_i = \text{rank}(H_{i,i-1})$, $i = 2, 3, \dots, k$. Assume also that B_{11} has n_1 columns.

We consider two cases. The algorithm comprises of implementing these two cases as the situations warrant. The feedback matrix F is obtained by accumulating feedback matrices from the individual cases.

Case 1. If $m_1 = n_1 - n_2 > 0$, that is, if $n_1 > n_2$, then we can immediately allocate $m_1 = n_1 - n_2$ eigenvalues as follows:

Write

$$\begin{aligned} \begin{pmatrix} H_{11} \\ H_{21} \end{pmatrix} &= \left(\begin{array}{c|c} H_{10} & * \\ \hline 0 & R_2 \end{array} \right), \\ B_{11} &= \begin{pmatrix} \hat{B}_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}. \end{aligned}$$

Then we have

$$H - \tilde{B}F = \left(\begin{array}{c|c} H_{10} & G_1 \\ \hline \cdots & \cdots \\ 0 & H_2 \end{array} \right) - \left(\begin{array}{c|c} \hat{B}_{11} & B_{12} \\ \hline 0 & B_{22} \\ \hline & 0 \end{array} \right) \left(\begin{array}{c|c} F_{11} & H_1 \\ \hline & F_2 \end{array} \right).$$

That is, the feedback matrix F_{11} for this allocation can be immediately found by solving

$$H_{10} - B_{11}F_{11} = \begin{pmatrix} \text{diag}(\lambda_1, \dots, \lambda_{m_1}) \\ 0 \end{pmatrix}.$$

Because of the last equation, we now have

$$H - \tilde{B}F = \left(\begin{array}{c|c} \text{diag}(\lambda_1, \dots, \lambda_m) & G_1 - \hat{B}_{11}H_1 \\ \hline & -B_{12}F_2 \\ \hline 0 & H_2 - B_2F_2 \end{array} \right)$$

where $B_2 = \begin{pmatrix} B_{22} \\ 0 \end{pmatrix}$.

Since B_{22} is a nonsingular upper triangular matrix and H_2 is still an unreduced upper Hessenberg matrix having the same form as H , (H_2, B_2) is a controllable pair.

We then solve the problem of finding F_2 such that $H_2 - B_2 F_2$ has the remaining eigenvalues. However, this time note that the first two blocks on the diagonal are $n_2 \times n_2$, thus, no more immediate assignment of eigenvalues is possible. The other eigenvalues have to be assigned using a different approach. If $n_2 = 1$, we then have a single-input problem to solve. Otherwise, we solve the multi-input problem with $n_1 = n_2$, using the approach below.

Case 2. Let $n_1 = n_2 = \dots = n_r > n_{r+1} \dots \geq n_k > 0$, for $1 < r \leq k$.

Suppose we want to assign an eigenvalue λ_1 to $H - \tilde{B}F$.

Then the idea is to find a unitary matrix Q_1 such that

$$Q_1^*(H - \tilde{B}F)Q_1 = \left(\begin{array}{c|c} \lambda_1 & * \\ \hline 0 & H_2 - B_2 F_2 \end{array} \right).$$

The unitary matrix Q_1 can be found as the product of the Givens rotations such that

$$(H - \lambda_1 I)Q_1 e_1 = \begin{pmatrix} a_1 \\ 0 \end{pmatrix}.$$

For example, if $n = 4, m = 2, k = 2$, and $n_1 = n_2 = 2$, then $r = 2$.

$$H - \lambda_1 I = \left(\begin{array}{cc|cc} * & * & * & * \\ * & * & * & * \\ \hline \textcircled{\text{*}} & * & * & * \\ 0 & \textcircled{\text{*}} & * & * \end{array} \right).$$

Then Q_1 is the product of two Givens rotations Q_{11} and Q_{21} , where Q_{11} annihilates the entry h_{42} and Q_{21} annihilates the entry h_{31} . Thus

$$(H - \lambda_1 I)Q_{11}Q_{21} = (H - \lambda_1 I)Q_1 = \left(\begin{array}{c|ccc} * & * & * & * \\ * & * & * & * \\ \hline 0 & * & * & * \\ 0 & 0 & * & * \end{array} \right).$$

Once Q_1 is found, F can be obtained by solving $B_{11}f_1 = a_1$, where $FQ_1 = (f_1, F_2)$, and $\tilde{B} = \begin{pmatrix} B_{11} \\ 0 \end{pmatrix}$.

Note that B_{11} is nonsingular.

It can now be shown that (H_2, B_2) is controllable and has the original form that we started with. The process can be continued to allocate the remaining eigenvalues with the pair (H_2, B_2) .

To summarize, the allocation of eigenvalues is done using unitary transformations when $n_1 = n_2$ or without unitary transformations when

$$n_1 > n_2.$$

(Note that the case 1 ($n_1 > n_2$) is a special case of case 2 with $Q_1 \equiv 1$, and $r = 1$).

Eventually, the process will end up with a single-input system which can be handled with a single-input algorithm described before.

For details of the process, see Miminis and Paige (1988).

Example 11.3.2 Let's consider Example 11.3.1 again. The eigenvalues to be assigned are: $\lambda_1 = 1, \lambda_2 = 2, \lambda_3 = 3, \lambda_4 = 4$ and $\lambda_5 = 5$.

Then

$$H_{11} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{pmatrix}, \quad H_{21} = (0 \ 0 \ 0 \ 1).$$

$$n_1 = 3, \quad n_2 = 1.$$

Since $m_1 = n_1 - n_2 = 2$, the two eigenvalues 1, and 2, can be assigned immediately as in Case 1.

$$H_{10} = \begin{pmatrix} 1 & 2 \\ 1 & 1 \\ 2 & 1 \end{pmatrix}, \quad B_{11} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{pmatrix}.$$

Solving for F_{11} from

$$H_{10} - B_{11}F_{11} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix}$$

$$\text{we have } F_{11} = \begin{pmatrix} -0.3333 & 3.3333 \\ -0.3333 & -1.6667 \\ 0.6667 & 0.3333 \end{pmatrix}$$

Deflation

$$H = \left(\begin{array}{cc|ccc} 1 & 2 & 3 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right) = \left(\begin{array}{c|c} H_{10} & G_1 \\ \hline 0 & H_2 \end{array} \right)$$

$$\tilde{B} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \left(\begin{array}{cc|c} 1 & 1 & 1 \\ 0 & 1 & 2 \\ \hline 0 & 0 & 3 \end{array} \right) = \left(\begin{array}{c|c} \hat{B}_{11} & B_{12} \\ \hline 0 & B_{22} \end{array} \right)$$

Then $H_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix}$, $B_2 \equiv B_{22} = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}$.

(H_2, B_2) is controllable.

Now we find F_2 such that $H_2 - B_2 F_2$ has the eigenvalues $(3, 4, 5)$.

This is a **single-input** problem. Using any of the single-input algorithms discussed before, we obtain

$$F_2 = (-3, 9.6667, -13.6667).$$

So, the required feedback matrix F is given by

$$F = \left(\begin{array}{c|c} F_{11} & 0 \\ \hline F_2 & \end{array} \right) = \left(\begin{array}{cc|ccc} -0.3333 & 3.33333 & 0 & 0 & 0 \\ -0.3333 & -1.6667 & 0 & 0 & 0 \\ \hline 0.6667 & 0.33333 & -3 & 9.6667 & -13.6667 \end{array} \right)$$

Verify: The eigenvalues of $H - \tilde{B}F$ are 1, 2, 5, 4, and 3.

The next Example shows the uses of both Case 1 and Case 2.

Example 11.3.3 Consider the controller-Hessenberg pair

$$H - \begin{pmatrix} 0.5828 & 0.4868 & -1.1540 & 0.2766 & 0.2554 \\ 0.0424 & 0.2132 & -0.9913 & 0.3373 & 0.4416 \\ -0.7890 & -0.7421 & 1.8240 & -0.9074 & -0.2173 \\ 0 & 0.8506 & -0.0109 & 0.6151 & 0.2352 \\ 0 & 0 & -0.3463 & -0.0639 & 0.4523 \end{pmatrix}$$

$$\tilde{B} = \begin{pmatrix} -0.4169 & 0.4149 & 1.0048 \\ 0 & 0.3815 & 0.9240 \\ 0 & 0 & -1.4043 \\ 0 & 0 & 0 \end{pmatrix}$$

Let the eigenvalues to be assigned be $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{0.0321, 0.7274, 1.0750, 1.3545, 3.7444\}$.

$$n_1 = 3, n_2 = 2.$$

Case 1. Immediate allocation of 1 eigenvalue ($\lambda_1 = 0.0321$).

$$F = \begin{pmatrix} -1.2103 & 0 & 0 & 0 & 0 \\ -1.2497 & 0 & 0 & 0 & 0 \\ 0.5619 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Deflation

$$H = H_2 = \begin{pmatrix} 0.2132 & -0.9913 & 0.3373 & 0.4416 \\ -0.7421 & 1.8240 & -0.9074 & -0.2173 \\ 0.8506 & -0.0109 & 0.6151 & 0.2352 \\ 0 & -0.3463 & -0.0639 & 0.4523 \end{pmatrix}.$$

$$\tilde{B} = B_2 = \begin{pmatrix} 0.3815 & 0.9240 \\ 0 & -1.4043 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} B_{22} \\ 0 \end{pmatrix}.$$

$$n_1 = n_2 = 2.$$

Thus no more immediate eigenvalue assignment is possible and we consider Case 2.

Case 2. Regular allocation of 1 eigenvalue ($\lambda_2 = 0.7224$)

$$H - \lambda_2 I = \begin{pmatrix} -0.5142 & -0.9913 & 0.3373 & 0.4416 \\ -0.7421 & 1.0966 & -0.9074 & -0.2173 \\ 0.8506 & -0.0109 & -0.1123 & 0.2352 \\ 0 & -0.3463 & -0.0639 & -0.2751 \end{pmatrix}$$

Eliminate entry (4, 2) of this matrix by rotation Q_{11}

Update $H - \lambda_2 I$:

$$(H - \lambda_2 I)Q_{11} = \begin{pmatrix} -0.5142 & -0.5116 & 0.9136 & 0.4416 \\ -0.7421 & 1.0913 & -0.9137 & -0.2173 \\ 0.8506 & 0.1084 & 0.0311 & 0.2352 \\ 0 & 0 & 0.3521 & -0.2751 \end{pmatrix}$$

Eliminate entry (3, 1) of this matrix by rotation Q_{21} .

Update $(H - \lambda_2 I)Q_{11}$:

$$(H - \lambda_2 I)Q_{11}Q_{21} = \begin{pmatrix} -0.4424 & -0.5748 & 0.9136 & 0.4416 \\ 1.1764 & -0.5981 & -0.9137 & -0.2173 \\ -0.0000 & 0.8574 & 0.0311 & 0.2352 \\ 0 & 0 & 0.3521 & -0.2751 \end{pmatrix}$$

Update the feedback matrix F :

$$F = \begin{pmatrix} -1.2103 & 0 & 0 & 0 \\ -1.2497 & 0.8692 & 0 & 0 \\ 0.5619 & -0.8377 & 0 & 0 \end{pmatrix}$$

$$Q_1 = Q_{11}Q_{21} = \begin{pmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0 & -0.1265 & 0.1800 & -0.9755 & 0 \\ 0 & 0.9920 & 0.0230 & -0.1244 & 0 \\ 0 & 0 & -0.9834 & -0.1815 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

Deflation again using $Q_{21}^T Q_{11}^T (H - \lambda_2 I) Q_{11} Q_{21}$. The reduced matrices are given by:

$$H = H_2 = \begin{pmatrix} 0.0369 & 0.8814 & 0.4038 \\ 0.4325 & 1.6203 & 0.1710 \\ 0 & 0.3521 & 0.4523 \end{pmatrix}$$

$$\tilde{B} = B_2 = \begin{pmatrix} 0.3784 & 0.8843 \\ 0 & 1.3810 \\ 0 & 0 \end{pmatrix}$$

$$n_1 = 2, n_2 = 1.$$

Since $n_1 > n_2$, we return again to the immediate placement of the eigenvalue $\lambda_3 = 1.0750$.

Case 1. Immediate allocation of the eigenvalue $\lambda_3 = 1.0750$

$$F \text{ (updated)} = \begin{pmatrix} -1.2103 & 0 & 0 & 0 \\ -1.2497 & 0.8692 & -3.4751 & 0 \\ 0.5619 & -0.8377 & 0.3132 & 0 \end{pmatrix}$$

Since \tilde{B} is now a column vector, we solve a **single-input problem** to place the remaining eigenvalues λ_4 and λ_5 :

$$H = \begin{pmatrix} 1.6203 & 0.1710 \\ 0.3521 & 0.4523 \end{pmatrix}$$

$$\tilde{B} = \begin{pmatrix} 1.3810 \\ 0 \end{pmatrix}, \quad \lambda_4 = 1.3545, \quad \lambda_5 = 3.7444$$

Updated feedback matrix F :

$$F = \begin{pmatrix} -1.2103 & 0 & 0 & 0 & 0 \\ -1.2497 & 0.8692 & -3.4751 & 0 & 0 \\ 0.5619 & -0.8377 & 0.3132 & 0.2243 & -6.6019 \end{pmatrix}$$

Flop-count. The solution of the Hessenberg multi-input problem, using the above described method requires about $\frac{23}{7}n^3$ flops.

When combined with about $6n^3$ flops required for the multi-input controller-Hessenberg reduction, the total count is about $9n^3$ flops.

Stability. The round-off error analysis performed by Miminis and Paige (1988) shows that the algorithm is **numerically backward stable**. Specifically, it can be shown that the computed feedback matrix K is such that

$$\Omega((A + \Delta A) - (B + \Delta B)K) = \Omega(L),$$

where $\|\Delta A\|$ and $\|\Delta B\|$ are small, and L is the matrix with eigenvalues $\lambda_i + \delta\lambda_i$, $i = 1, \dots, n$; where $|\delta\lambda_i| \leq |\lambda_i|\mu$, μ is the machine precision.

Avoiding Complex Arithmetic

The method as described above might give a complex feedback matrix; because, it is an explicit shift algorithm. To avoid complex arithmetic to assign complex conjugate pairs, the idea of implicit shift and the double step need to be used.

MATCONTROL NOTE: The explicit QR algorithm has been implemented in MATCONTROL function **poleqr.m**.

11.3.3 The Schur Method for the Multi-input Eigenvalue Assignment Problem

As the title suggests the following algorithm due to A. Varga (1981) for the multi-input EVA is based on the reduction of the matrix A to the **real Schur form** (RSF). So, unlike the other two methods just described, the Schur method does not follow the “Template.”

Let $R = QAQ^T = \begin{pmatrix} A_1 & A_3 \\ 0 & A_2 \end{pmatrix}$ be the real Schur form (RSF) of A and let $\hat{B} = QB$ be the transformed control matrix.

Let's partition \hat{B} as

$$\hat{B} = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}.$$

Then, since (A, B) is controllable, so is (A_2, B_2) .

Suppose that the real Schur form R of A has been **ordered** in such a way that A_1 contains the “**good**” eigenvalues and A_2 contains the “**bad**” ones. The “**good**” eigenvalues are the ones we want to retain and the “**bad**” ones are those we want to reassign.

It is, therefore, natural to ask how the feedback matrix F can be determined such that after the application of feedback, the eigenvalues of A_1 will remain unchanged, while those in A_2 will be changed to “desired” ones by feedback.

The answer to this question is simple. If the feedback matrix F is taken in the form $F = (0, F_2)$, then after the application of the feedback matrix to the pair (R, \hat{B}) we have

$$R - \hat{B}F = \begin{pmatrix} A_1 & A_3 - B_1F_2 \\ 0 & A_2 - B_2F_2 \end{pmatrix}.$$

This shows that the eigenvalues of $R - \hat{B}F$ are the union of the eigenvalues of A_1 and of $A_2 - B_2F_2$.

The problem thus reduces to finding F_2 such that $A_2 - B_2F_2$ has a desired spectrum.

The special structure of A_2 now can be exploited.

Since the diagonal blocks of A_2 are either scalars (1×1) or 2×2 matrices, all we then need a procedure to assign eigenvalues to a $p \times p$ matrix where $p = 1$ or 2.

The following is a simple procedure to do this.

Algorithm 11.3.2 An Algorithm to Assign p ($p = 1$, or 2) Eigenvalues

Inputs: M - The state matrix of order p
 G - The control matrix of order $p \times m$
 Γ_p - The set of p complex numbers, closed under complex conjugation.
 r - Rank of G .

Output: F_p - The feedback matrix such that $(M - GF_p)$ has the spectrum Γ_p .

Assumption: (M, G) is controllable.

Step 1. Find the SVD of G ; that is, find U and V such that $G = U(\hat{G}, 0)V^T$, where \hat{G} is $r \times r$.

Step 2. Update M : $\hat{M} = U^T M U$.

Step 3. If $r = p$, compute $\hat{F}_p = (\hat{G})^{-1}(\hat{M} - J)$, where J is $p \times p$ and the eigenvalues of J are the set Γ_p . Go to step 6.

Step 4. Let $\Gamma_2 = \{\lambda_1, \lambda_2\}$ and

$$\hat{M} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}, \hat{G} = \begin{pmatrix} \beta \\ 0 \end{pmatrix}.$$

Step 5. Compute $\hat{F}_p = (\hat{F}_{p1}, \hat{F}_{p2})$ as follows:

$$\begin{aligned} \hat{F}_{p1} &= (m_{11} + m_{22} - \lambda_1 - \lambda_2) / \beta \\ \hat{F}_{p2} &= \left(\frac{m_{22}}{m_{21}} \right) \hat{F}_{p1} - (m_{11}m_{22} - m_{12}m_{21} - \lambda_1\lambda_2) / (m_{21}\beta) \end{aligned}$$

Step 6. Compute $F_p = V \begin{bmatrix} \hat{F}_p \\ 0 \end{bmatrix} U^T$.

Example 11.3.4

Consider applying the algorithm with the following data:

$$M = \begin{pmatrix} -0.8416 & 0.5614 \\ -0.9695 & -1.2195 \end{pmatrix}, \quad G = \begin{pmatrix} 0.0399 & -0.0495 & 0.1874 \\ 0.5067 & 0.4985 & -1.0846 \end{pmatrix}, \quad \Gamma_p = \{3, 2\}.$$

Step 1.

$$U = \begin{pmatrix} 0.1235 & -0.9923 \\ -0.9923 & -0.1236 \end{pmatrix}$$

$$\hat{G} = \begin{pmatrix} 1.3067 & 0 \\ 0 & 0.1153 \end{pmatrix}$$

$$V = \begin{pmatrix} -0.3810 & -0.8862 \\ -0.3833 & -0.1081 \\ 0.8414 & -0.4506 \end{pmatrix}$$

Step 2.

$$\hat{M} = \begin{pmatrix} -1.1637 & -1.0096 \\ 0.5213 & -0.8974 \end{pmatrix}$$

Step 3.

$$\hat{F}_p = \begin{pmatrix} -2.4211 & -0.7726 \\ 4.5203 & -33.7951 \end{pmatrix}$$

Step 6.

$$F_p = \begin{pmatrix} -30.392 & -0.6769 \\ -3.8657 & -0.9238 \\ -14.9682 & 2.2416 \end{pmatrix}.$$

Verify: The eigenvalues of $M - GF_p$ are 3 and 2.

Algorithm 11.3.2 can now be used in an iterative fashion to assign all the eigenvalues of A_2 , by shifting only 1 or 2 eigenvalues at a time.

The process starts with the last $p \times p$ block of A_2 and then after the assignment with this block is completed using the algorithm above, a new $p \times p$ diagonal block is moved, using orthogonal similarity, in the last diagonal position, and the assignment procedure is repeated on this new block.

The required feedback matrix is the sum of component feedback matrices, each of which assigns 1 or 2 eigenvalues.

The overall procedure then can be summarized as follows:

Algorithm 11.3.3 The Schur Algorithm for the Multi-input EVA Problem

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix

Γ - The set of numbers to be assigned, closed under complex conjugation

Output: K - The feedback matrix such that the numbers in the set Γ belong to the spectrum of $A - BK$.

Assumption: (A, B) is controllable.

Step 1. Transform A to the ordered real Schur form:

$$A \equiv Q A Q^T = \begin{bmatrix} A_1 & A_3 \\ 0 & A_2 \end{bmatrix},$$

where A_1 is $r \times r$, A_2 is $(n-r) \times (n-r)$; A_1 contains the “good” eigenvalues and A_2 contains the “bad” eigenvalues.

Update $B \equiv QB$ and set $\hat{Q} = Q$.

Step 2. Set $K \equiv 0$ (zero matrix), and $i = r + 1$. If $i > n$, stop.

Step 3. Set M equal to the last block in A of order p ($p = 1$ or 2) and set G equal to the last p rows of B .

Step 4. Compute F_p using Algorithm 11.3.2 to shift p eigenvalues from the set Γ .

Step 5. Update K and A : $K \equiv K - (0, F_p)\hat{Q}$, $A \equiv A - B(0, F_p)$.

Step 6. If $i \geq n - p$, stop.

Step 7. Move the last block of A in position (i, i) accumulating the transformations in Q , and update $B \equiv QB$, and $\hat{Q} = Q\hat{Q}$.

Step 8. Set $i \equiv i + p$ and go to step 3.

Remarks.

1. The ordering of the real Schur form in Step 1 has to be done according to the procedure described in Chapter 4.
2. It has been tacitly assumed that “the complex numbers in Γ are chosen and ordered so that the ordering agrees with the diagonal structure of the matrix A_2 .” If this requirement is not satisfied, some interchange of the blocks of A_2 need to be done so that the required condition is satisfied, using an appropriate orthogonal similarity [Exercise 12].
3. The final matrix K at the end of this algorithm is the sum of the component feedback matrices, each of them assigning 1 or 2 eigenvalues.
4. The algorithm has the additional flexibility to solve a “**partial eigenvalue assignment problem**”, which concerns reassigning only the “bad” eigenvalues, leaving the “good” ones unchanged.

Example 11.3.5

Let's apply Algorithm 11.3.3 with data from Example 11.3.1.

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The eigenvalues of A are: $\{-1.0305, -0.7131i, -1.0305 + 0.7131i, 0.2805, 1.8369, 4.9437\}$

Step 1.

$$A = QAQ^T = \begin{pmatrix} -0.4543 & 1.0893 & -0.2555 & -0.7487 & -0.5053 \\ -0.7717 & -1.6068 & 0.3332 & -1.2007 & 2.6840 \\ -0.0000 & -0.0000 & 0.2805 & -0.2065 & 0.2397 \\ -0.0000 & -0.0000 & -0.0000 & 1.8369 & -3.1302 \\ 0.0000 & 0.0000 & 0.0000 & -0.0000 & 4.9437 \end{pmatrix}$$

$$Q = \begin{pmatrix} -0.2128 & 0.0287 & 0.6509 & -0.6606 & 0.3064 \\ 0.8231 & -0.1628 & -0.2533 & -0.3926 & 0.2786 \\ 0.1612 & -0.8203 & 0.4288 & 0.2094 & -0.2708 \\ -0.4129 & -0.4850 & -0.3749 & 0.0539 & 0.6714 \\ 0.2841 & 0.2539 & 0.4332 & 0.6023 & 0.5517 \end{pmatrix}$$

Let the desired closed-loop eigenvalues be the same as in Example 11.3.1.: $S = \{5, 4, 3, 2, 1\}$

Updating $B = QB$:

$$B = \begin{pmatrix} -0.2128 & -0.1841 & 1.7974 \\ 0.8231 & 0.6603 & -0.2625 \\ 0.1612 & -0.6591 & -0.1929 \\ -0.4129 & -0.8979 & -2.5077 \\ 0.2841 & 0.5380 & 2.0916 \end{pmatrix}$$

Step 2.

$$K = 0, i = 1.$$

Step 3.

$$p = 1$$

$$M = (4.9437)$$

$$G = (0.2841, 0.5380, 2.0916)$$

$$G_p = 5$$

Step 4.

$$F_p = \begin{pmatrix} -0.0034 \\ -0.0064 \\ -0.0248 \end{pmatrix}$$

Step 5. Updated K and A are:

$$K = \begin{pmatrix} -0.0010 & -0.0009 & -0.0015 & -0.0020 & -0.0019 \\ -0.0018 & -0.0016 & -0.0028 & -0.0038 & -0.0035 \\ -0.0071 & -0.0063 & -0.0108 & -0.0150 & -0.0137 \end{pmatrix}$$

$$A = \begin{pmatrix} -0.4543 & 1.0893 & -0.2555 & -0.7487 & -0.4626 \\ -0.7717 & -1.6068 & 0.3332 & -1.2007 & 2.6845 \\ -0.0000 & -0.0000 & 0.2805 & -0.2065 & 0.2313 \\ -0.0000 & -0.0000 & -0.0000 & 1.8369 & -3.1996 \\ 0.0000 & 0.0000 & 0.0000 & -0.0000 & 5.0000 \end{pmatrix}$$

Step 7. Reorder A and update \hat{Q} and B :

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -1.1307 & -2.4841 \\ 0 & 1.8369 & 0.1125 & -1.2452 & -0.4711 \\ 0 & 0 & 0.2805 & -0.0392 & -0.4337 \\ 0 & 0 & 0 & -0.5524 & -1.2822 \\ 0 & 0 & 0 & 0.5749 & -1.5087 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0.1065 & 0.3723 & 0.0603 & -0.6543 & 0.6468 \\ -0.4437 & 0.0466 & -0.1060 & 0.5961 & 0.6591 \\ -0.0986 & 0.1525 & 0.9746 & 0.1263 & -0.0346 \\ -0.8654 & 0.1928 & -0.0846 & -0.3388 & -0.3033 \\ 0.1823 & 0.8938 & -0.1676 & 0.2930 & -0.2325 \end{pmatrix}$$

$$\hat{Q} = \begin{pmatrix} 0.7474 & 0.3745 & 0.5264 & 0.1504 & 0.0376 \\ 0.0568 & -0.0551 & -0.2840 & 0.6816 & 0.6696 \\ 0.2416 & -0.8972 & 0.2528 & 0.1953 & -0.1859 \\ 0.3829 & 0.1005 & -0.6528 & 0.2773 & -0.5833 \\ 0.4828 & -0.2040 & -0.3902 & -0.6307 & 0.4187 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.7474 & 1.1219 & 3.0756 \\ 0.0568 & 0.0017 & -0.9056 \\ 0.2416 & -0.6556 & -0.7945 \\ 0.3829 & 0.4834 & -1.3744 \\ 0.4828 & 0.2789 & -1.0956 \end{pmatrix}$$

Step 8. $i = 2$ and repeat the Step 3 - Step 7.

$$\begin{aligned} p &= 2 \\ M &= \begin{pmatrix} 0.5524 & -1.2822 \\ 0.5749 & -1.5087 \end{pmatrix} \\ G &= \begin{pmatrix} 0.3829 & 0.4834 & -1.3744 \\ 0.4828 & 0.2789 & -1.0956 \end{pmatrix} \\ G_p &= (2, 1) \end{aligned}$$

Step 4.

$$F_p = \begin{pmatrix} 7.0705 & -6.3576 \\ -5.1373 & 3.4356 \\ 1.7303 & 0.7260 \end{pmatrix}$$

Step 5.

$$\begin{aligned} K &= \begin{pmatrix} -0.3630 & 2.0061 & -2.1362 & 5.9678 & -6.7883 \\ -0.3103 & -1.2184 & 2.0102 & -3.5949 & 4.4318 \\ 1.0061 & 0.0194 & -1.4235 & 0.0069 & -0.7191 \end{pmatrix} \\ A &= \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & -3.8198 \\ 0 & 1.8369 & 0.1125 & -0.0712 & 0.5417 \\ 0 & 0 & 0.2805 & -3.7408 & 3.9316 \\ 0 & 0 & 0 & 1.6016 & 0.4896 \\ 0 & 0 & 0 & 0.4896 & 1.3984 \end{pmatrix} \end{aligned}$$

Step 7. Reorder A and update \hat{Q} and B .

$$\begin{aligned} A &= \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & -3.8198 \\ 0 & 1.0000 & 0.3087 & 0.1527 & -5.2208 \\ 0 & 0 & 2.0000 & 0.2457 & 1.2394 \\ 0 & 0.0000 & 0.0000 & 1.8369 & -0.8889 \\ 0 & 0 & 0 & 0 & 0.2805 \end{pmatrix} \\ Q &= \begin{pmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0.2022 & -0.9708 & 0.0815 & -0.1001 \\ 0 & 0.8160 & 0.1647 & 0.4035 & 0.3797 \\ 0 & -0.5415 & -0.1144 & 0.6385 & 0.5348 \\ 0 & 0 & -0.1318 & -0.6503 & 0.7482 \end{pmatrix} \\ \hat{Q} &= \begin{pmatrix} 0.7474 & 0.3745 & 0.5264 & 0.1504 & 0.0376 \\ -0.2402 & 0.8885 & -0.3169 & 0.0340 & 0.2265 \\ 0.4240 & -0.2297 & -0.6017 & 0.4608 & 0.4394 \\ 0.4443 & 0.0875 & -0.5006 & -0.5517 & -0.4899 \\ 0.0804 & -0.0997 & 0.0993 & -0.6779 & 0.7171 \end{pmatrix} \end{aligned}$$

$$B = \begin{pmatrix} 0.7474 & 1.1219 & 3.0756 \\ -0.2402 & 0.6483 & 0.5859 \\ 0.4240 & 0.1944 & -1.8404 \\ 0.4443 & 0.5319 & -0.8823 \\ 0.0804 & -0.0193 & 0.1787 \end{pmatrix}$$

Step 8. $i = 4$ and repeat Step 3-Step 7.

$$p = 1$$

$$M = 0.2805$$

$$G = (0.0804 \ -0.0193 \ 0.1787)$$

$$G_p = 4$$

Step 4.

$$F_p = \begin{pmatrix} -7.7091 \\ 1.8532 \\ -17.1422 \end{pmatrix}$$

Step 5.

$$K = \begin{pmatrix} -0.9827 & 2.7748 & -2.9014 & 11.1937 & -12.3165 \\ -0.1613 & -1.4032 & 2.1942 & -4.8511 & 5.7607 \\ -0.3719 & 1.7286 & -3.1250 & 11.6272 & -13.0117 \end{pmatrix}$$

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & 52.5851 \\ 0 & 1.0000 & 0.3087 & 0.1527 & 1.7698 \\ 0 & 0 & 2.0000 & 0.2457 & -27.4012 \\ 0 & 0.0000 & 0.0000 & 1.8369 & -13.5735 \\ 0 & 0 & 0 & 0 & 4.0000 \end{pmatrix}$$

Step 7. Reorder A and update \hat{Q} and B

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & 52.5851 \\ 0 & 1.0000 & 0.3087 & 0.1527 & 1.7698 \\ 0 & 0 & 2.0000 & 0.2457 & -27.4012 \\ 0 & 0.0000 & 0.0000 & 4.0000 & -13.5735 \\ 0 & 0 & 0 & 0 & 1.8369 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0.9875 & -0.1574 \\ 0 & 0 & 0 & 0.1574 & 0.9875 \end{pmatrix}$$

$$\hat{Q} = \begin{pmatrix} 0.7474 & 0.3745 & 0.5264 & 0.1504 & 0.0376 \\ -0.2402 & 0.8885 & -0.3169 & 0.0340 & 0.2265 \\ 0.4240 & -0.2297 & -0.6017 & 0.4608 & 0.4394 \\ 0.4261 & 0.1021 & -0.5099 & -0.4382 & -0.5966 \\ 0.1493 & -0.0847 & 0.0192 & -0.7563 & 0.6311 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.7474 & 1.1219 & 3.0756 \\ -0.2402 & 0.6483 & 0.5859 \\ 0.4240 & 0.1944 & -1.8404 \\ 0.4261 & 0.5283 & -0.8994 \\ 0.1493 & 0.0646 & 0.0377 \end{pmatrix}$$

Step 8. $i = 5$ and repeat Step 3 - Step 7.

$$p = 1$$

$$M = 1.8369$$

$$G = (0.1493 \ 0.0646 \ 0.0377)$$

$$G_p = 3$$

Step 4.

$$F_p = \begin{pmatrix} -6.2271 \\ -2.6950 \\ -1.5710 \end{pmatrix}$$

Step 5.

$$K = \begin{pmatrix} -1.9124 & 3.3022 & -3.0213 & 15.9029 & -16.2462 \\ -0.5637 & -1.1750 & 2.1423 & -2.8130 & 4.0599 \\ -0.6064 & 1.8617 & -3.1553 & 12.8153 & -14.0031 \end{pmatrix}$$

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & 65.0948 \\ 0 & 1.0000 & 0.3087 & 0.1527 & 2.9415 \\ 0 & 0 & 2.0000 & 0.2457 & -27.1285 \\ 0 & 0.0000 & 0.0000 & 4.0000 & -10.9093 \\ 0 & 0 & 0 & 0 & 3.0000 \end{pmatrix}$$

and the algorithm ends.

Verification: the eigenvalues of $A - BK$ are:

$$\begin{pmatrix} 5.000000000000024 \\ 3.99999999999960 \\ 1.000000000000000 \\ 3.000000000000018 \\ 1.999999999999999 \end{pmatrix}$$

Flop-count and Stability. The algorithm requires about $30n^3$ flops, most of which is consumed in the reduction of A to the real Schur form and ordering of this real Schur form.

The algorithm is believed to be numerically stable (note that it is based on all numerically stable operations). **However, no formal round-off error analysis of the algorithm has been performed yet.**

MATCONTROL function: Algorithm 11.3.3 has been implemented in MATCONTROL function **polesch**.

11.4 Conditioning of the Feedback Problem

In this section, we will discuss the sensitivity of the feedback problem, that is, **we are interested in determining a measure that describes how small perturbations in the data affect the computed feedback.** We discuss the **single-input case** first.

11.4.1 The Single-Input Case

The discussion here has been taken from the Ph.D. Thesis of Mark Arnold (1993).

Consider the map

$$\Phi : (\mathbb{R}^{n \times n} \times \mathbb{R}^n \times \mathbb{C}^n) \longrightarrow \mathbb{R}^n \quad (11.4.1)$$

given by

$$\Phi(A, b, \Omega) = f,$$

then roughly speaking, the condition of Φ at (A, b, Ω) is the radius of the image of a δ -ball about (A, b, Ω) .

We present the conditioning of the Hessenberg single input feedback problem only. Recall that in the Hessenberg case, the **explicit formula** for f (Equation 11.2.9) is

$$f^T = \alpha e_n^T \phi(H). \quad (11.4.2)$$

Thus, the **Hessenberg single-input eigenvalue assignment problem is essentially a polynomial evaluation problem at an unreduced Hessenberg matrix.** We therefore first present a condition number for evaluating a polynomial matrix, and then show how it can be used to identify condition numbers for the feedback problem.

The condition of the map

$$\tilde{\Phi} : (H, \Omega) \longrightarrow e_n^T \phi(H),$$

where $\phi(x) = (x - \lambda_1)(x - \lambda_2) \cdots (x - \lambda_n)$ is developed as in Rice (1966). We define the set

$$\mathcal{H}_\delta = \{H + E : \|E\| \leq \delta \|H\|\},$$

and call the radius of its image under $\tilde{\Phi}$

$$r(\mathcal{H}_\delta) = \sup_{\|E\| \leq \delta \|H\|} \|e_n^T \phi(H + E) - e_n^T \phi(H)\| / \delta. \quad (11.4.3)$$

We can also define a relative radius

$$s(\mathcal{H}_\delta) = \sup_{\|E\| \leq \delta \|H\|} \frac{\|e_n^T \phi(H + E) - e_n^T \phi(H)\|}{\delta \|e_n^T \phi(H)\|}. \quad (11.4.4)$$

The condition numbers κ and ν are then defined as

$$\kappa(H) = \lim_{\delta \rightarrow 0} r(\mathcal{H}_\delta), \quad (11.4.5)$$

and

$$\nu(H) = \lim_{\delta \rightarrow 0} s(\mathcal{H}_\delta). \quad (11.4.6)$$

If we denote the Fréchet derivative of ϕ at the point H , by $D(\phi(H))$, then when it exists, it satisfies

$$\phi(H + E) = \phi(H) + D(\phi(H))E + O(\|E\|). \quad (11.4.7)$$

Theorem 11.4.1 *The Fréchet derivative $D(\phi(H))$ acting on E is given by*

$$D(\phi(H))E = \sum_{i=1}^n \phi_i(H, E), \quad (11.4.8)$$

where

$$\phi_i(H, E) = \left[\prod_{j=1}^{i-1} (H - \lambda_j I) \right] E \left[\prod_{j=i+1}^n (H - \lambda_j I) \right]. \quad (11.4.9)$$

■

Let $b^T = \beta e_1^T = \beta(1, 0, 0)^T$.

Consider now the map: $\Phi : (H, b) \rightarrow \alpha e_n^T \phi(H)$. Define now

$$\hat{\nu} = \sup_{\|E\|, |\gamma| \leq 1} (\|H\| + |\beta|) \|\alpha e_n^T D(\phi(H))E - \alpha \left(\sum_{i=1}^n \frac{\gamma_i}{\beta_i} \right) e_n^T \phi(H)\| / \|\alpha e_n^T \phi(H)\|, \quad (11.4.10)$$

and

$$\hat{\kappa} = \sup_{\|E\|, |\gamma| \leq 1} \|\alpha e_n^T D(\phi(H))E - \alpha \left(\sum_{i=1}^n \frac{\gamma_i}{\beta_i} \right) e_n^T \phi(H)\|. \quad (11.4.11)$$

Then, it has been shown in Arnold (1993) that these are the condition numbers for single-input Hessenberg feedback problem. One can expect that for small ϵ , perturbations on the order of ϵ in H and β will lead to relative (absolute) errors in the feedback bounded

above by $\hat{\nu}$ ($\hat{\kappa}$). For simplicity, and without much loss of information, we will use the following quantities in place of $\hat{\nu}$ and $\hat{\kappa}$:

$$\nu = \sup_{\|E\|, |\gamma| \leq 1} (\|H\| \|e_n^T D(\phi(H)) E\| + \frac{\|\beta\| \left\| (\sum_{i=1}^n \frac{\gamma_i}{\beta_i}) e_n^T \phi(H) \right\|}{\|e_n^T \phi(H)\|}), \quad (11.4.12)$$

and

$$\kappa = \sup_{\|E\|, |\gamma| \leq 1} |\alpha| (\|e_n^T D(\phi(H)) E\| + \left\| (\sum_{i=1}^n \frac{\gamma_i}{\beta_i}) e_n^T \phi(H) \right\|), \quad (11.4.13)$$

where $\beta_i = h_{i,i-1}$, $i = 2, 3, \dots, n$.

Bounds for the Condition Numbers

Arnold (1993) has obtained various bounds for the above condition numbers. The following Theorem summarizes some of his important results.

Theorem 11.4.2

$$\nu(H, \beta) \leq |\beta| \cdot \|w\| + \frac{\|H\| \|D(\phi(H))\|}{\|e_n^T \phi(H)\|}, \quad (11.4.14)$$

and

$$\kappa(H, \beta) \leq |\alpha| (\|D(\phi(H))\| + \|w\| \|e_n^T \phi(H)\|), \quad (11.4.15)$$

where $w = (\frac{1}{\beta}, \frac{1}{h_{21}}, \dots, \frac{1}{h_{n,n-1}})^T$.

Theorem 11.4.3

$$|\beta|^2 + \frac{\|H\| \|e_n^T \phi'(H)\|}{\|e_n^T \phi(H)\|} \leq \nu(H, \beta) \leq |\beta| \cdot \|w\| + \frac{\|H\| \|D(\phi(H))\|}{\|e_n^T \phi(H)\|}, \quad (11.4.16)$$

and

$$|\alpha| \left(\|e_n^T \phi'(H)\| + \frac{\|e_n^T \phi(H)\|}{|\beta|} \right) \leq \kappa(H, \beta) \leq |\alpha| (\|D(\phi(H))\| + \|w\| \|e_n^T \phi(H)\|). \quad (11.4.17)$$

Proof: See Arnold (1993).

Estimating the Condition Numbers of the Feedback Problem

Based on the above results, Arnold (1993) has identified four estimates of $\nu(H, \beta)$ out of which the following

$$\nu_{d\phi} = \frac{|\beta| \|\omega\| + \|H\| \|e_n^T \phi'(H)\|}{\|e_n^T \phi(H)\|} \quad (11.4.18)$$

where $\omega = (\frac{1}{\beta}, \frac{1}{h_{21}}, \dots, \frac{1}{h_{n,n-1}})^T$, has worked very well in test examples. Defining the quantity digits off (as in Rice (1966)) by

$$\log_{10} \left[\left(\frac{\mu \nu_{\text{estimate}}}{\text{err}} \right) \right], \quad (11.4.19)$$

where μ is the machine precision ($\mu \approx 2 \times 10^{-16}$), then it has been shown that the maximum digits off in estimating conditioning for 100 ill-conditioned problems are only 1.86, and minimum digits off are 0.51. **Thus it never underestimated the error and overestimated the error by less than 2 digits.**

The computation of this condition estimator requires only $\frac{2n^3}{3}$ flops once the system is in controller-Hessenberg form. For details of these experiments, see Arnold (1993). **Note these bounds work only for the single-input Hessenberg EVA problem.**

11.4.2 The Multi-input Case

We have just considered a perturbation analysis of the single-input Hessenberg feedback problem by considering only the perturbation of the matrix H and the vector b .

Sun (1996) has studied perturbation analysis of the both single-input and multi-input problems by allowing perturbations of all the data; namely A, B , and $S = \{\lambda_1, \dots, \lambda_n\}$. Below, we state his result for the multi-input problem, without proof. The proof can be found in Sun (1996).

Let $\lambda_i \neq \lambda_j$, for all $i \neq j$. Let $K = (k_1, \dots, k_m)^T$, and $X = (x_1, \dots, x_n)$ be such that

$$A + BK = X\Lambda X^{-1}, \quad (11.4.20)$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$.

Also, let y_1, \dots, y_n be the normalized left eigenvectors of $A + BK$; that is, $Y = X^{-T} = (y_1, \dots, y_n)$, which implies $y_i^T x_j = \delta_{ij}$ for all i and j .

Suppose that the data matrices A, B and Λ and the feedback matrix K are so perturbed that the resulting closed-loop matrix has also the distinct eigenvalues.

Let $B = (b_1, \dots, b_m)$. Define now

$$W_k = (S_1 X^T, S_2 X^T, \dots, S_m X^T)_{n \times mn} \quad (11.4.21)$$

where $S_j = \text{diag}(y_1^T b_j, \dots, y_n^T b_j)$, $j = 1, 2, \dots, m$.

Also define

$$W_a = (D_1(X)X^{-1}, D_2(X)X^{-1}, \dots, D_n(X)X^{-1})_{n \times n^2} \quad (11.4.22)$$

$$W_b = \text{diag}(T_1 X^{-1}, T_2 X^{-1}, \dots, T_m X^{-1})_{n \times nm} \quad (11.4.23)$$

and $W_\lambda = -I_n$,

where

$$D_i(X) = \text{diag}(x_{i1}, \dots, x_{in}), \quad i = 1, \dots, n \quad (11.4.24)$$

$$T_j = \text{diag}(k_j^T x_1, \dots, k_j^T x_n), \quad j = 1, \dots, m, \quad (11.4.25)$$

and $x_i = (x_{i1}, \dots, x_{in})$.

$$\text{Also, let } Z = W_k^\dagger, \Phi = -ZW_a, \text{ and } \Psi = -ZW_b. \quad (11.4.26)$$

Here W_k^\dagger denotes the generalized inverse of W_k .

Theorem 11.4.4 (Perturbation Bound for a Multi-input Feedback Matrix)

Suppose that the controllable pair (A, B) is slightly perturbed to another controllable pair (\tilde{A}, \tilde{B}) , and that the self-conjugate set $S = \{\lambda_1, \dots, \lambda_n\}$, $\lambda_i \neq \lambda_j, i \neq j$ is slightly perturbed to the set $\tilde{S} = \{\tilde{\lambda}_1, \dots, \tilde{\lambda}_n\}$, $\tilde{\lambda}_i \neq \tilde{\lambda}_j, i \neq j$.

Let K be the feedback matrix of the EVA problem with the data A, B, S . Then there is a solution \tilde{K} to the problem with data \tilde{A}, \tilde{B} , and \tilde{S} such that for any consistent norm $\|\cdot\|$, we have

$$\begin{aligned} \|\tilde{K} - K\| &\leq \delta_K + O\left(\left\|\begin{pmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{\lambda} \end{pmatrix} - \begin{pmatrix} a \\ b \\ \lambda \end{pmatrix}\right\|^2\right) \\ &\leq \Delta_K + O\left(\left\|\begin{pmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{\lambda} \end{pmatrix} - \begin{pmatrix} a \\ b \\ \lambda \end{pmatrix}\right\|^2\right), \end{aligned} \quad (11.4.27)$$

where

$$\begin{aligned} a &= \text{vec}(A), \quad \tilde{a} = \text{vec}(\tilde{A}), \\ b &= \text{vec}(B), \quad \tilde{b} = \text{vec}(\tilde{B}), \\ \lambda &= (\lambda_1, \dots, \lambda_n)^T, \quad \tilde{\lambda} = (\tilde{\lambda}_1, \tilde{\lambda}_2, \dots, \tilde{\lambda}_n)^T, \\ \delta_K &= \|\Phi(\tilde{a} - a) + \Psi(\tilde{b} - b) + Z(\tilde{\lambda} - \lambda)\|, \\ \Delta_K &= \|\Phi\| \|\tilde{a} - a\| + \|\Psi\| \|\tilde{b} - b\| + \|Z\| \|\tilde{\lambda} - \lambda\|, \end{aligned} \quad (11.4.28)$$

Z, Ψ , and Φ are defined by (11.4.26).

11.4.3 Absolute and Relative Condition Numbers

The three groups of **absolute condition numbers** that reflect the three different types of sensitivities of K with respect to the data A, B , and S , respectively, have been obtained by Sun (1996). These are:

$$\kappa_A(K) = \|\Phi\|, \quad \kappa_B(K) = \|\Psi\|, \quad \text{and } \kappa_\lambda(K) = \|Z\|. \quad (11.4.29)$$

Furthermore the scalar $\kappa(K)$ defined by

$$\kappa(K) = \sqrt{\kappa_A^2(K) + \kappa_B^2(K) + \kappa_\lambda^2(K)} \quad (11.4.30)$$

can be regarded as an **absolute condition number** of K .

If $\|\cdot\|_2$ is used, then

$$\kappa_A(K) = \|\Phi\|_2 \leq \|Z\| \|X^{-1}\|_2, \quad (11.4.31)$$

$$\kappa_B(K) = \|\Psi\|_2 \leq \max_{1 \leq j \leq n} \|k_j\|_2 \|Z\|_2 \|X^{-1}\|_2, \quad (11.4.32)$$

$$\kappa_\lambda(K) = \|Z\|_2. \quad (11.4.33)$$

Thus, using the Frobenius norm, the respective **relative condition numbers** are given by

$$\kappa_A^{(r)}(K) = \kappa_A(K) \frac{\|A\|_F}{\|K\|_F}, \quad (11.4.34)$$

$$\kappa_B^{(r)}(K) = \kappa_B(K) \frac{\|B\|_F}{\|K\|_F}, \quad (11.4.35)$$

and

$$\kappa_\lambda^{(r)}(K) = \kappa_\lambda(K) \frac{\|\lambda\|_2}{\|K\|_F}, \quad \lambda = (\lambda_1, \dots, \lambda_n)^T. \quad (11.4.36)$$

Furthermore, the **relative condition number of K** is given by

$$\kappa_K^{(r)}(K) = \sqrt{(\kappa_A^{(r)}(K))^2 + (\kappa_B^{(r)}(K))^2 + (\kappa_\lambda^{(r)}(K))^2}, \quad (11.4.37)$$

where $\kappa_A^{(r)}(K)$, $\kappa_B^{(r)}(K)$, and $\kappa_\lambda^{(r)}(K)$ are evaluated using 2-norm.

Remarks: A variation of the above results appears in Mehrmann and Xu (1996, 1997)), where it has been shown that the ill-conditioning of the feedback problem is also related to the ill-conditioning of the open-loop eigenvector matrix and the distance to uncontrollability (see next section for more on this).

Example 11.4.1 (*Sun (1996), Laub and Linnemann (1986)*).

Consider the following single-input problem:

$$A = \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ \alpha & -3 & 0 & 0 & 0 \\ 0 & \alpha & -2 & 0 & 0 \\ 0 & 0 & \alpha & -1 & 0 \\ 0 & 0 & 0 & \alpha & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

and

$$S = \{-2.9992, -0.8808, -2, -1, 7.0032 \times 10^{-14}\}.$$

Choose $\alpha = 0.0010$.

Then $K = (3.12, -1.67, 7.45, -2.98, 0.37)$.

The feedback problem with the above data is expected to be ill-conditioned, as $\kappa_A^{(r)}(K) = 3.2969 \times 10^{12}$, $\kappa_B^{(r)}(K) = 1.01117 \times 10^{12}$, $\kappa_\lambda^{(r)}(K) = 2.3134 \times 10^{12}$ and $\kappa_K^{(r)}(K) = 4.1527 \times 10^{12}$.

Indeed, if only the 1st entry of S is changed to -3 and all other data remain unchanged, then the feedback vector for this perturbed problem becomes $\hat{K} = (3.1192, 0.0078, 7.8345, 0.0004, 0.3701)$.

11.5 Conditioning of the Closed-loop Eigenvalues

Suppose that the feedback matrix K has been computed using a stable algorithm, that is the computed feedback matrix \hat{K} is the exact feedback matrix for a nearby EVA problem. The question now is: **How far are the eigenvalues of the computed closed loop matrix $\hat{M}_c = A - BK$ from the desired eigenvalues $\{\lambda_1, \dots, \lambda_n\}$?** Unfortunately, the answer to this question is: **Even though a feedback matrix has been computed using a numerically stable algorithm, there is no guarantee that the eigenvalues of the closed-loop matrix will be near those which are to be assigned.**

The following interrelated factors, either individually, or in combination, can contribute to the conditioning of the closed-loop eigenvalues:

- The conditioning of the problem of determining the feedback matrix K from the given data.
- The condition number (with respect to a p -norm) of the eigenvector matrix of the closed-loop system.
- The distance to uncontrollability, and the distance between the open-loop and closed-loop eigenvalues.
- The norm of the feedback matrix.

Regarding the first factor, we note that if the problem of computing the feedback matrix is ill-conditioned, then even with the use of a stable numerical algorithm, the computed feedback matrix cannot be guaranteed to be accurate, and, as a result, the computed closed-loop eigenvalues might differ significantly from those to be assigned. (Note that the eigenvalue problem of a nonsymmetric matrix can be very ill-conditioned).

The relation of the other two factors to the conditioning of the closed-loop eigenvalues can be explained by the following analysis using the **Bauer-Fike Theorem (Chapter 3)**.

Let $M_c = A - BK$ and let $E = \hat{M}_c - M_c$, where $\hat{M}_c = A - B\hat{K}$, \hat{K} being the computed value of K .

Let X be the transforming matrix that diagonalizes the matrix M_c ; that is $X^{-1}M_cX = \text{diag}(\lambda_1, \dots, \lambda_n)$. Let μ be an eigenvalue of \hat{M}_c . Then, by the Bauer-Fike Theorem (**Theorem 3.3.3**), we have

$$\min_{\lambda_i} |\lambda_i - \mu| \leq \text{Cond}_2(X) \|E\|_2.$$

Again, $E = A - B\hat{K} - (A - BK) = B(K - \hat{K}) = B\Delta K$

Thus, we see that the product of the spectral condition number of X and the $\|B\Delta K\|_2$ influences the distance between the desired poles and those obtained with a computed \hat{K} .

This again is related to the factors: norm of the computed feedback matrix \hat{K} , distance to the uncontrollability of the pair (A, B) , and the distance between closed-loop poles and the eigenvalues of A , etc. (See Mehrmann and Xu (1997), Theorem 4.1). Note that some of these

observations also follow from the explicit formula of the feedback vector (11.2.7) in the single-input case, and the one in the multi-input case derived in Arnold (1993).

Example 11.5.1 Consider EVA with the following data:

$$A = \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 0.0001 & -3 & 0 & 0 & 0 \\ 0 & 0.0001 & -2 & 0 & 0 \\ 0 & 0 & 0.0001 & -1 & 0 \\ 0 & 0 & 0 & 0.0001 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

$$S = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{10, 12, 24, 29, 30\}.$$

$$\text{Then } K = (-115, 4.887 \cdot 10^7, -9.4578 \cdot 10^{12}, 8.1915 \cdot 10^{17}, -2.5056 \cdot 10^{22})$$

The eigenvalue assignment problem with the above data is very ill-conditioned as the following computation shows.

Change the entry a_{51} of A to 10^{-6} and keep all other data unchanged. The eigenvalues of the closed-loop matrix then become: $\{\pm 1.5829 \cdot 10^8, -3, -2, -1\}$.

The explanation of this drastic change in the closed-loop eigenvalues can be given in the light of the discussions we just had in the last section.

- (i) **Ill-conditioning of the feedback vector:** Let \hat{K} be obtained by changing the first entry of K to -114.999 and leaving the remaining entries unchanged.

The eigenvalues of $(A - B\hat{K})$ are $\{29.5386 \pm 0.4856j, 23.9189, 12.0045, 9.9984\}$.

So, the problem of computing the feedback vector K is ill-conditioned.

- (ii) All the subdiagonal entries of A are small, indicating that the system is near an uncontrollable system.
- (iii) **Distance between the open-loop and closed-loop eigenvalues:** The open-loop eigenvalues are $\{0, -1, -2, -3, -4\}$.

Thus, the open-loop eigenvalues are well-separated from those of the closed-loop eigenvalues.

- (iv) **Ill-conditioning of the eigenvector matrix:** $\text{Cond}_2(X) = 1.3511 \times 10^{24}$.

Thus, the spectral condition-number of the closed-loop matrix is large. The condition numbers of the individual eigenvalues are also large.

Note: In Example 11.5.1, the feedback vector K was computed using Algorithm 11.2.1. The MATLAB function **place** cannot place the eigenvalues.

Concluding Remarks

We have identified several factors that contribute to the ill-conditioning of the closed-loop eigenvalues.

In general, the problem of assigning eigenvalues is an intrinsically ill-conditioned problem. Indeed, in He, Laub, and Mehrmann (1995), it has been shown that in the single-input case, the feedback vector K (which is unique) depends upon the solution of a linear system whose matrix is a **Cauchy matrix** [Exercise 16], and a Cauchy matrix is well-known to be ill-conditioned for large order matrices. Furthermore, recent studies by (Mehrmann and Xu (1998), and Calvetti, Lewis, and Reichel (1999) have shown that the **distribution of eigenvalues is also an important factor for conditioning of the eigenvalue-assignment problem**, and the condition number of the problem can be reduced by choosing the eigenvalues judiciously in a prescribed compact set in the complex plane. For details, see the above papers.

11.6 Robust Eigenvalue Assignment

In the last section we have discussed the aspect of the closed-loop eigenvalue sensitivity due to perturbations in the data A , B , and K .

The problem of finding a feedback matrix K such that the closed-loop eigenvalues are as insensitive as possible is called the **robust eigenvalue assignment (REVA)** problem.

Several factors affecting the closed-loop eigenvalue sensitivity were identified in the last section; the principal of those factors being the conditioning of the closed-loop eigenvector matrix.

In this section, we consider REVA with respect to minimizing the condition number of the eigenvector matrix of the closed-loop matrix.

In the multi-input case, one can think of solving the problem by making use of the available freedom. One such a method was proposed by Kautsky, Nichols and Van Dooren (1985). For an excellent account of the robust eigenvalue assignment problem and discussion on this and other methods, see the paper by Byers and Nash (1989). See also Tits and Yang (1996).

11.6.1 Measures of Sensitivity

Let the matrix $M = A - BK$ be diagonalizable; that is, assume that there exists a nonsingular matrix X such that

$$X^{-1}(A - BK)X = \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n).$$

Recall from Chapter 3 (see also Wilkinson (1965), Datta (1995), etc.) that a measure of sensitivity c_j of an individual eigenvalue λ_j due to perturbations in the data A , B , and K is given by $c_j = \frac{1}{s_j} = \frac{\|y_j\|_2 \|x_j\|_2}{|y_j^T x_j|}$, where x_j and y_j are, respectively, the right and left eigenvectors of M corresponding to λ_j . Furthermore, the overall sensitivity of all the eigenvalues of the matrix M is given by $\text{Cond}_2(X) = \|X\|_2 \|X^{-1}\|_2$. Note also that $\max_j c_j \leq \text{Cond}_2(X)$.

Thus, two natural measures of sensitivity are:

$$\begin{aligned}\nu_1 &= \| C \|_\infty \\ \nu_2 &= \text{Cond}_2(X),\end{aligned}$$

where $C = (c_1, c_2, \dots, c_n)^T$.

One could also take (see Kautsky, Nichols, and Van Dooren (1985))

$$\nu_3 = \| X^{-1} \|_F n^{\frac{1}{2}} = \| C \|_2 n^{\frac{1}{2}}$$

$$\text{and } \nu_4 = (\sum_j \sin^2 \theta_j)^{\frac{1}{2}} / n^{\frac{1}{2}}$$

as other measures. Here θ_j are the angles between the eigenvectors x_j and certain corresponding orthonormal vectors $\hat{x}_j, j = 1, 2, \dots, n$.

11.6.2 Statement of the Robust Eigenvalue Assignment Problem

In view of the above, the *robust eigenvalue assignment problem* with respect to minimizing the conditioning of the eigenvalue matrix X can be formulated as follows:

Given $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m} (m \leq n)$, having full rank, and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$, find a real matrix K and a nonsingular matrix X satisfying

$$(A - BK)X = X\Lambda. \quad (11.6.1)$$

such that some measures ν of the sensitivity of the closed-loop eigenproblem is optimized.

The following result, due to Kautsky, Nichols and Van Dooren (1985), gives conditions under which a given nonsingular X can be assigned to (11.6.1).

Theorem 11.6.1 *Given a nonsingular X , and Λ as above, there exists a matrix K satisfying (11.6.1) if and only if*

$$U_1^T (X\Lambda - AX) = 0, \quad (11.6.2)$$

where U_1 is defined by

$$B = [U_0, U_1] \begin{bmatrix} Z \\ 0 \end{bmatrix}, \quad (11.6.3)$$

with $U = [U_0, U_1]$ orthogonal and Z nonsingular.

The matrix K is explicitly given by

$$K = Z^{-1} U_0^T (A - X\Lambda X^{-1}). \quad (11.6.4)$$

Proof: Since B has full rank, the factorization of B given by (11.6.3) exists with Z nonsingular. Again, from (11.6.1) we have

$$BK = A - X\Lambda X^{-1} \quad (11.6.5)$$

Multiplying (11.6.5) by U^T , we obtain

$$\begin{aligned} ZK &= U_0^T(A - X\Lambda X^{-1}) \\ 0 &= U_1^T(A - X\Lambda X^{-1}). \end{aligned} \quad (11.6.6)$$

Since X and Z are invertible, we immediately have (11.6.2) and (11.6.4). ■

11.6.3 A Solution Technique for the Robust Eigenvalue Assignment Problem

Theorem 11.6.1 suggests the following algorithm for a solution of the robust eigenvalue assignment problem.

Algorithm 11.6.1 A Robust Eigenvalue Assignment Algorithm (The KNV Algorithm)

Input: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix with full rank

Λ - The diagonal matrix containing the eigenvalues $\lambda_1, \dots, \lambda_n$

Assumptions: (A, B) is controllable and $\lambda_1, \dots, \lambda_n$ is a self-conjugate set

Output: K - The feedback matrix such that the spectrum of $A - BK$ is the set $\{\lambda_1, \dots, \lambda_n\}$, and the condition number of the eigenvector matrix is as small as possible.

Step 1. Decompose the matrix B to determine U_0 , U_1 , and Z as in (11.6.3).

Construct orthonormal bases, comprised by the columns of matrices S_j and \hat{S}_j for the space $s_j = N\{U_1^T(A - \lambda_j I)\}$ and its complement \hat{s}_j , for $\lambda_j \in s_j, j = 1, 2, \dots, n$.

Step 2. Select a set of n normalized vectors x_1, \dots, x_n from the space s_j such that $X = (x_1, \dots, x_n)$ is well-conditioned.

Step 3. Compute $M = A - BK$ by solving the linear systems: $MX = X\Lambda$.

Step 4. Compute K : $K = Z^{-1}U_0^T(A - M)$.

Some Implementational Details

Implementation of Step 1.

Decomposition of B in Step 1 of the algorithm amounts to the QR factorization of B . Once this decomposition is performed, constructions of the bases can be done either by QR factorization of $(U_1^T(A - \lambda_j I))^T$ or by computing its SVD.

If QR factorization is used, then

$$(U_1^T(A - \lambda_j I))^T = (\hat{S}_j, S_j) \begin{pmatrix} R_j \\ 0 \end{pmatrix}.$$

Thus S_j and \hat{S}_j are the matrices whose columns form the required orthonormal bases.

If SVD is used, then from

$$U_1^T(A - \lambda_j I) = T_j (\Gamma_j, 0)(\hat{S}_j, S_j)^T,$$

we see that the columns of S_j and \hat{S}_j form the required orthonormal bases. Here Γ_j is the diagonal matrix containing the singular values.

Note: The QR decomposition, as we have seen, is more efficient than the SVD approach.

Implementation of Step 2.

Step 2 is the key step in the solution process. Kautsky, Nichols and Van Dooren (1985) have proposed four methods to implement Step 2. Each of these four methods aims at minimizing a different measure ν of the sensitivity.

We present here only one (Method 0 in their paper), which is the simplest and most natural one. This method is designed to minimize the measure $\nu_2 = \text{Cond}_2(X)$.

First, we note that $\text{Cond}_2(X)$ will be minimized if each vector $x_j \in s_j$, $j = 1, 2, \dots, n$ is chosen such that the angle between x_j and the space

$$t_j = \langle x_i, i \neq j \rangle$$

is maximized for all j . The symbol $\langle x_k \rangle$ denotes the space spanned by the vectors x_k .

This can be done in an iterative fashion. Starting with an **arbitrary** set of n independent vectors $X = (x_1, \dots, x_n)$, $x_j \in s_j$, $j = 1, \dots, n$, we replace each vector x_j by a new vector such that the angle to the current space t_j is maximized for each j . The QR method is again used to compute the new vectors as follows:

Find \tilde{y}_j by computing the QR decomposition of

$$\begin{aligned} X_j &= (x_1, x_2, \dots, x_{j-1}, x_{j+1}, \dots, x_n) \\ &= (\tilde{Q}_j, \tilde{y}_j) \begin{pmatrix} \tilde{R}_j \\ 0 \end{pmatrix}. \end{aligned}$$

and then compute the new vector

$$x_j = \frac{S_j S_j^T \tilde{y}_j}{\|S_j^T \tilde{y}_j\|_2}.$$

Note that with this choice of x_j , the condition $c_j = 1/|\tilde{y}_j^T x_j|$ is minimized. The n steps of the process required to replace successively n vectors x_1 through x_n will constitute a sweep.

At the end of each sweep, $\text{Cond}_2(X)$ is measured to see if it is acceptable; if not, a new iteration is started with the current X as the starting set. The iteration is continued until $\text{Cond}_2(X)$, after a full sweep of the powers ($j = 1, 2, \dots, n$), is less than some positive tolerance.

Implementations of Step 3 and Step 4

Implementations of Step 3 and Step 4 are straightforward. M in Step 3 is computed by solving linear systems:

$$X^T M^T = \Lambda^T X^T$$

using Gaussian elimination with partial pivoting.

K in Step 4 is computed by solving upper triangular systems:

$$ZK = U_0^T(A - M).$$

Flop-Count

Step 1: $O(n^3m)$ flops.

Step 2: $O(n^3) + O(n^2m)$ flops per *sweep*.

Step 3: $O(n^3)$ flops.

Step 4: $O(mn^2)$ flops.

MATCONTROL NOTE: Algorithm 11.6.1 has been implemented in MATCONTROL function **polerob**. It computes both the feedback matrix K and the transforming matrix X .

Remarks on Convergence of Algorithm 11.6.3 and the Tits-Yang Algorithm

Each step of the above iteration amounts to rank-one updating of the matrix X such that the sensitivity of the eigenvalue λ_j is minimized. **However, this does not necessarily mean that the overall conditioning is improved at each step.** This is because the conditioning of the other eigenvalues ($\lambda_i, i \neq j$) will be disturbed when the old vector x_j is replaced by the new vector.

It was, thus, stated by Kautsky, Nichols, and Van Dooren (1985) that “the process does not necessarily converge to a fixed point.” It, however, turned out to be the case of “slow convergence” only. Indeed, Tits and Yang (1996) later gave a proof of the convergence of the algorithm. Tits and Yang (1996) observed that the this algorithm amounts to maximize, at each iteration, the determinant of the eigenvector matrix X with respect to one of its column (subject to the constraints that it is still an eigenvector matrix of the closed-loop system). Based on this observation, Tits and Yang developed a more efficient algorithm by maximizing $\det(X)$ with respect to two columns concurrently. The **Tits-Yang algorithm** is easily extended to

assign the eigenvalues with complex conjugate pairs. For details of these algorithms we refer the readers to the paper of Tits and Yang (1996). There also exists a software called **robpole** based on the Tits-Yang algorithm.

Example 11.6.1 Consider the robust eigenvalue assignment with the following data:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, B = \begin{pmatrix} 6 & 3 \\ 1 & 2 \\ 8 & 9 \end{pmatrix}, \Lambda = \text{diag}(9, 5, 1).$$

Step 1.

$$U_0 = \begin{pmatrix} -0.5970 & 0.7720 \\ -0.0995 & -0.3410 \\ -0.7960 & -0.5364 \end{pmatrix}, U_1 = \begin{pmatrix} -0.2181 \\ -0.9348 \\ 0.2804 \end{pmatrix}$$

$$Z = \begin{pmatrix} -10.0499 & -9.1543 \\ 0 & -3.1934 \end{pmatrix}.$$

$$\text{Step 2. } X = \begin{pmatrix} -0.0590 & 0.9859 & -0.0111 \\ -0.7475 & 0.0215 & -0.8993 \\ -0.6617 & -0.1657 & 0.4371 \end{pmatrix}$$

$$\text{Step 3. } M = \begin{pmatrix} 5.0427 & 0.0786 & 0.2640 \\ 0.9987 & 3.7999 & 5.7856 \\ 0.1399 & 2.0051 & 6.1575 \end{pmatrix}$$

$$\text{Step 4. } K = \begin{pmatrix} -1.8988 & 0.0269 & 0.5365 \\ 2.4501 & 0.5866 & -0.1611 \end{pmatrix}.$$

Verify: The eigenvalues of $(A - BK)$ are 5, 9, 1.

$$\text{Cond}_2(K) = 6.3206.$$

MATLAB Note: The MATLAB function **place** has implemented Algorithm 11.6.1.

Given a controllable pair (A, B) and a vector p containing the eigenvalues to be assigned, $K = \text{place}(A, B, p)$ computes the feedback matrix K such that $(A - BK)$ has the desired eigenvalues. The software **robpole**, based on the Tits-Yang algorithm, can be obtained from the authors.

Some Properties of the Closed-Loop System

The minimization of the condition number of the eigenvector matrix leads to some desirable robust properties of the closed-loop system. We state some of these properties below. The proofs can be found in Kautsky et al. (1985) or the readers can work out the proofs themselves.

Theorem 11.6.2 (i) The gain matrix K obtained by Algorithm 11.6.1 satisfies the inequality

$$\|K\|_2 \leq (\|A\|_2 + \max_j |\lambda_j| \text{Cond}_2(X)) / (\sigma_{\min}(B)) = k',$$

where $\sigma_{\min}(B)$ denotes the smallest singular value of B .

(ii) The transient response $x(t)$ satisfies

$$\|x(t)\|_2 \leq \text{Cond}_2(X) \max_j \{|e^{\lambda_j t}|\} \cdot \|x_0\|_2,$$

where $x(0) = x_0$ or in the discrete case

$$\|x(k)\|_2 \leq \text{Cond}_2(X) \cdot \max_j \{|\lambda_j|^k \cdot \|x_0\|_2\}.$$

■

Example 11.6.2

For Example 11.6.1, we easily see that

$$\begin{aligned}\|K\|_2 &= 3.2867 \\ k' &= 12.8829\end{aligned}$$

Thus the result of Part (i) of Theorem 11.6.2 is verified.

Theorem 11.6.3 If the feedback matrix K assigns a set of stable eigenvalues λ_j , then the perturbed closed-loop matrix $A - BK + \Delta$ remains stable for all perturbations Δ that satisfy

$$\|\Delta\|_2 \leq \min_{s=j\omega} \sigma_{\min}(sI - A + BK) = \delta(K).$$

Furthermore $\|\delta(K)\| \geq \min_j \text{Re}(-\lambda_j) / \text{Cond}_2(X)$.

In the discrete-case, the closed-loop system remains stable for perturbations Δ such that

$$\|\Delta\|_2 \leq \min_{s=\exp(i\omega)} \{sI - A + BK\} = \Delta(F)$$

and $\Delta(F) \geq \min_j (1 - |\lambda_j|) / \text{Cond}_2(X)$.

■

Minimum-Norm Robust Pole Assignment

As stated in the previous section that the norm of the feedback matrix is another important factor that influences the sensitivity of closed-loop poles. Thus, it is important to consider this aspect of robust eigenvalue assignment as well.

The robust eigenvalue assignment with respect to minimizing the norm of the feedback matrix has been considered by Keel, Fleming, and Bhattacharyya (1985) and more recently by Varga (2000).

Both algorithms are Sylvester equation based (see **Exercise 14** for the statement of a Sylvester equation based eigenvalue assignment algorithm). The paper by Keel et al. (1985) addresses minimization of the performance index

$$I - \text{Trace}(K^T K)$$

whereas Varga (2000) considers the minimization of the performance index

$$J = \frac{\alpha}{2} (\| X \|_F^2 + \| X^{-1} \|_F) + \frac{1-\alpha}{2} \| K \|_F^2$$

Note that minimizing J as above for $0 < \alpha < 1$ amounts to simultaneous minimization of the norm of the feedback matrix K and of the condition number of the eigenvector matrix X (with respect to the Frobenius norm).

For space limitations, we are not able to describe these algorithms here. The readers are referred to the papers of Keel et al. (1985) and Varga (2000).

11.7 Table of Comparison for Efficiency and Stability: Single-input EVA Problem

Method	Efficiency: Flop-Count (Approximate) This count includes transformation of (A, b) to the Controller-Hessenberg form.	Numerical Stability (Backward Stability and Other Features)
The Recursive Algorithm (Algorithm 11.2.1)	$\frac{11}{3}n^3$	Stability is not guaranteed , but the algorithm allows the users to monitor the stability. Reliable .
The RQ implementations of the recursive algorithm (Algorithms 11.2.2 and 11.2.3)	$5n^3$	Stable .
The explicit QR method (Exercise 9)	$5n^3$	Stable
The implicit QR method (Exercise 10)	$5n^3$	Stable
The eigenvector method (Petkov, Christov and Konstantinov (1984); Exercise 8).	$\frac{20}{3}n^3$.	Stable

11.8 Table of Comparison for Efficiency and Stability: Multi-input EVA Problem

Method	Efficiency: Flop-Count (Approximate). These counts include transformation of (A, B) to the Controller-Hessenberg form.	Numerical Stability (Backward Stability) and other Features
The Recursive Algorithm (Algorithm 11.3.1)	$\frac{19}{3}n^3$	No formal round-off error analysis available. The algorithm is believed to be reliable .
The explicit QR algorithm (Section 11.3.2).	$9n^3$	Stable.
The implicit QR algorithm (The multi-input version of the implicit single-input QR algorithm of Patel and Misra (1984)).	$9n^3$	Stability not formally proven, but is believed to be stable .
The Schur Method (Algorithm 11.3.3)	$30n^3$	Stability not formally proven, but is believed to be stable . The algorithm has an attractive feature that it can also be used for partial pole placement in the sense that it allows one to reassign only the “bad” eigenvalues, leaving the “good” ones unchanged.
The eigenvector method (Petkov et al. (1986); not described in the book)	$\frac{40}{3}n^3$	Stable

11.9 Comparative Discussion of Various Methods and Recommendation

For the single-input problem: The recursive algorithm (**Algorithm 11.2.1**) is the fastest one proposed so far. It is also extremely simple to implement. Unfortunately, the numerical stability of the algorithm cannot be guaranteed in all cases. The algorithm, however, allows the users to monitor the stability. **Algorithm 11.2.1 is thus reliable.** In a variety of test examples, this algorithm has done remarkably well, even for some ill-conditioned problems (e.g. see **Example 11.2.4**). The RQ implementation of the recursive algorithm (**Algorithm 11.2.3**), the explicit and implicit QR algorithms (**Exercise 9 and Exercise 10**) all have the same efficiency, and are numerically **stable**. The eigenvector algorithm (**Exercise 8**) if properly implemented, is also stable, but it is the most expensive one of all the single-input algorithms. One important thing to note here is that there exist RQ implementations of all the single input algorithms mentioned in the Table 11.7 (Arnold and Datta (1998)). These RQ implementations are much easier to understand and implement on computers. **We strongly recommend the use of RQ implementations of these algorithms.**

For the multi-input problem: The recursive algorithm (**Algorithm 11.3.1**) is again the fastest algorithm; however, no round-off stability analysis of this algorithm has been done yet. **The explicit QR algorithm described in Section 11.3.2 is stable.** The properly implemented eigenvector algorithm due to Petkov, Christov and Konstantinov (1986), is also stable but is more expensive than the explicit QR algorithm. The Schur algorithm (**Algorithm 11.3.3**) is the most expensive one. It is believed to be numerically stable. An important feature of this algorithm is that it can be used for partial pole assignment in the sense that it offers a choice to the user to place only the "bad" eigenvalues, leaving the "good" ones unchanged.

The robust eigenvalue assignment algorithm (**Algorithm 11.6.1**) exploits the freedom offered by the problem to minimize the conditioning of the eigenvector matrix which is a major factor for the sensitivity of the closed-loop poles. However, when a well-conditioned eigenvector matrix does not exist, the algorithm may give inaccurate results. When the eigenvector matrix is ill-conditioned, it may be possible to obtain more accurate results using other methods.

*Based on the above observations, it is recommended that for the single-input problem, the recursive algorithm (**Algorithm 11.2.1**) be tried first. In case of possible ill-conditioning of the matrix L , its RQ formulation (**Algorithm 11.2.2**) should be used.*

*For the multi-input problem, the multi-input version of the recursive algorithm (**Algorithm 11.3.1**) should be tried first. If the algorithm appears to be unstable (as indicated by the condition number of the matrix L), the explicit QR algorithm (**Section 11.3.1**) is to be used. It should, however, be noted that Algorithm 11.3.1 and the explicit QR algorithm, as stated here, might give complex feedback matrix. There now exists an Arnoldi-based modified version of Algorithm 11.3.1 (Carvalho and Datta (2001)) that avoid complex arithmetic and this modified version has been implemented in MATCONTROL function **polercm**.*

For robust eigenvalue assignment and partial eigenvalue assignment, the choices are Algorithm

11.6.1 (or the Tits-Yang algorithm) and the Schur algorithm (**Algorithm 11.3.3**), respectively.

11.10 Some Selected Software

11.10.1 MATLAB CONTROL SYSTEM TOOLBOX

Classical design tools.

acker	- SISO pole placement
place	- MIMO pole placement

11.10.2 MATCONTROL

POLERCS	- Single-input pole placement using the recursive algorithm
POLEQRS	- Single-input pole placement using the QR version of the recursive algorithm
POLERQS	- Single-input pole placement using RQ version of the recursive algorithm
POLERCM	- Multi-input pole placement using the recursive algorithm
POLERCX	- Multi-input pole placement using the modified recursive algorithm that avoids complex arithmetic and complex feedback.
POLEQRM	- Multi-input pole placement using the explicit QR algorithm
POLESCH	- Multi-input pole placement using the Schur decomposition
POLEROB	- Robust pole placement

11.10.3 CSP-ANM

Pole assignment

- The recursive algorithm is implemented as `StateFeedbackGains [system, poles, Method → Recursive]`.
- The explicit QR algorithm is implemented as `StateFeedbackGains [system, poles, Method → QRDecomposition]`.
- The Schur method is implemented as `StateFeedbackGains [system, poles, Method → SchurDecomposition]`.
- The RQ implementation of the recursive single-input algorithm is implemented as `StateFeedbackGains [system, poles, Method → RecursiveRQDecomposition]`.
- The implicit single-input RQ algorithm is implemented as `StateFeedbackGains [system, poles, Method → ImplicitRQDecomposition]`.

11.10.4 SLICOT

Eigenvalue/Eigenvector Assignment

- SB01BD Pole assignment for a given matrix pair (A, B)
- SB01DD Eigenstructure assignment for a controllable matrix pair (A, B) in orthogonal canonical form
- SB01MD State feedback matrix of a time-invariant single-input system

11.10.5 MATRIX_X

Purpose: Calculate state feedback gains via pole placement for single input continuous-time or discrete-time systems.

Syntax: KC=POLEPLACE (A, B, POLES) ... controller design
 KE=POLEPLACE (A', B', POLES) ... estimator design

11.10.6 POLEPACK

A collection of MATLAB programs for eigenvalue assignment, developed by G.S. Miminis (1991). Available on **NETLIB**.

11.11 Summary and Review

- I. **Statement of the Eigenvalue Assignment Problem (The EVA Problem).** Given a pair of matrices (A, B) , and the set $S = \{\lambda_1, \dots, \lambda_n\}$, closed under complex conjugation, find a matrix K such that $\Omega(A - BK) = S$.

Here $\Omega(M)$ denotes the spectrum of M .

In the single-input case, the problem reduces to that of finding a row vector f^T such that

$$\Omega(A - bf^T) = S.$$

- II. **Existence and Uniqueness.** The EVA problem has a solution if and only if (A, B) is controllable. In the single-input case, the feedback vector, when it exists, is unique. In the multi-input case, when there exists a feedback matrix, there are many. Therefore, the existing freedom can be exploited to improve the conditioning of the solution and of the closed-loop eigenvectors.

- III. **Numerical Methods.** There are many methods for the EVA problem. Only a few have been described here. These include

- (i) Recursive algorithms (**Algorithm 11.2.1** for the single-input problem and **Algorithm 11.3.1** for the multi-input problem).

- (ii) QR-type algorithms (**Algorithms 11.2.2, 11.2.3 and those stated in Exercise 9 and Exercise 10** for the single-input problem and the explicit QR method described in **Section 11.3.2** for the multi-input problem).
- (iii) The Schur algorithm (**Algorithm 11.3.3**) for the multi-input problem.

IV. Efficiency and Numerical Stability. The recursive algorithms are the most efficient algorithms. The computer implementations of these algorithms are extremely simple. The algorithms, however, do not have guaranteed numerical stability; except for the RQ version of the single-input recursive algorithm, which has been proved to be numerically stable (Arnold and Datta (1998)).

In the single-input case, it has been proved (see Arnold and Datta (1998)), by forward round-off error analysis, that the stability of the recursive algorithm (**Algorithm 11.2.1**) can be monitored and it is possible for the user to know exactly when the algorithm starts becoming problematic. **It is thus reliable.** Similar results are believed to hold for the multi-input recursive algorithm as well. But no formal analysis in the multi-input case has yet been done.

The QR-type Algorithms for single-input problems all have the same efficiency and are numerically stable.

For the multi-input problem, the Schur Algorithm (**Algorithm 11.3.3**) is the most expensive one. However, it has an important feature; namely, it allows one to place only the “bad” eigenvalues, leaving the “good” ones unchanged.

The explicit QR algorithm (**Section 11.3.2**) and the multi-input version of the single-input implicit QR algorithm (not described in this book) have the same efficiency. The explicit QR algorithm has been proven to be stable and the implicit QR algorithm is believed to be stable as well.

V. Explicit Solutions. An explicit expression for the unique feedback vector for the single-input EVA problem has been given using the recursive algorithm (**Algorithm 11.2.1**). This formula is

$$f = \frac{1}{\alpha} (H^T - \lambda_1 I)(H^T - \lambda_2 I) \dots (H^T - \lambda_n I) e_n,$$

where $H = (h_{ij})$ is the Hessenberg matrix of the controller-Hessenberg form of the pair (A, b) and

$$\alpha = \prod_{i=1}^{n-1} h_{i+1,i}.$$

In the multi-input case, the expression is rather complicated (see Arnold (1993)).

1. Conditioning of the Feedback Problem: From the explicit expression of the feedback vector f of the single-input EVA problem, it is clear that the Hessenberg single-input feedback problem is essentially a polynomial evaluation $\phi(H)$ at an unreduced Hessenberg

matrix, where $\phi(x) = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_n)$, is the characteristic polynomial of the closed-loop matrix.

A result on the Frechet derivative $D\phi(H)$ is first given (**Theorem 11.4.1**), and the condition numbers for the feedback problem are then defined using this derivative. Next, a condition number estimator (**11.4.18**) for the problem is stated. It worked well on test examples. This estimator never underestimated the error and overestimated the error by less than 2 digits, in all 100 test examples of sizes varying from 10 to 50, both for ill-conditioned and well-conditioned problems.

In the multi-output case, **Theorem 11.4.4** gives the perturbation bound (11.4.27) for the feedback matrix from which the absolute and relative condition numbers are defined (**Section 11.4.3**). Specifically, the relations (11.4.29) and (11.4.30) define the absolute condition numbers and (11.4.34)-(11.4.37) define the relative condition numbers of the multi-input feedback problem.

VI. Conditioning of the Closed-Loop Eigenvalues. The major factors responsible for the sensitivity of the closed-loop eigenvalues have been identified in **Section 11.5**. These factors are: **the condition number of the eigenvector matrix of the closed-loop system, the distance to uncontrollability and the distance between the open-loop and the closed-loop eigenvalues, the conditioning of the feedback problem, and the norm of the feedback matrix.** The most important of them is the condition number of the eigenvector matrix.

VII. Robust Eigenvalue Assignment. Given the pair (A, B) and the matrix $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$, the problem is to find a nonsingular matrix X , and a matrix K satisfying

$$(A + BK)X = X\Lambda$$

such that $\text{Cond}_2(X)$ is minimum. In view of the last sentence of the preceding paragraph, the robust eigenvalue assignment problem with respect to minimizing the condition number of the eigenvector matrix is a very important practical problem. An algorithm (**Algorithm 11.6.1**) due to Kautsky, Nichols and Van Dooren (1985) is given in Section 11.6. The algorithm requires constructions of orthonormal bases for a certain space and for its complement. The QR factorization or the SVD can be used for this purpose. An analysis of convergence and a more improved version of this algorithm can be found in Tits and Yang (1996).

VIII. Algorithm for Minimizing Feedback Norm. Our discussions on the conditioning of the closed-loop eigenvalues (**Section 11.5**) show that it is also important to have algorithms that minimize the norm of the feedback matrix.

For such algorithms, see Keel, Fleming and Bhattacharyya (1985) and Varga (2000).

11.12 Chapter Notes and Further Reading

Many algorithms have been developed for solving the EVA by state-feedback. A good account of these algorithms can be found in the recent book by Xu (1998).

The earlier algorithms, based on reduction to controller-canonical forms, turns out to be numerically unstable. For a reference of some of these earlier algorithms, see Miminis and Paige (1982, 1988).

For a comprehensive reference of the Hessenberg or controller-Hessenberg based algorithms, which are more numerically reliable, see the recent paper of Arnold and Datta (1998). For a matrix equation based algorithm for eigenvalue assignment, see Bhattacharyya and DeSouza (1982). For robust eigenvalue and eigenstructure assignment algorithms, see Byers and Nash (1989), Kautsky, Nichols and Van Dooren (1985), and Tits and Yang (1996), Cavin and Bhattacharyya (1983). For robust eigenvalue assignment by output feedback, see Chu, et al. (1984) and references there in.

For algorithms that minimize the norm of the feedback matrix, see Keel, Fleming and Bhattacharyya (1985) and Varga (2000). The EVA problem by output feedback is a difficult problem and only a few algorithms available. See Misra and Patel (1989) for output feedback algorithms.

For the EVA and eigenstructure algorithms for descriptor systems (not discussed in this chapter), see Chu (1988), Fletcher, Kautsky and Nichols (1986), and Kautsky, Nichols and Chu (1989), Varga (2000). For partial pole assignment algorithms, see Varga (1981) and Saad (1988), Datta and Saad (1991).

For round-off error analysis of various algorithms for EVA, see Cox and Moss (1989, 1992), Arnold and Datta (1998), Miminis and Paige (1988).

The perturbation analysis for the single-input feedback problem has been taken from Arnold (1993) and that for the multi-input feedback problem has been taken from Sun (1996).

For discussions on conditioning of the EVA problem, see He, Laub and Mehrmann (1995), Mehrmann and Xu (1996, 1997, 1998), Konstantinov and Petkov (1993), Calvetti, Lewis and Reichel (1999).

For an extension of the single-input recursive algorithm to assigning Jordan canonical form, companion and Hessenberg forms, etc., see Datta and Datta (1990). For parallel algorithms for the EVA problem, see Bru et al. (1994c), Coutinho et al. (1995), Datta and Datta (1986), Datta (1991), Bakshi, Datta, and Roy (1994).

There also now exists a block algorithm (Carvalho and Datta (2001)) for the multi-input EVA. **This block algorithm, besides being suitable for highperformance computing, is guaranteed to give a real feedback matrix.**

EXERCISES

1. Modify both the single-input (**Algorithm 11.2.1**) so that the use of complex arithmetic can be avoided (consult Carvalho and Datta (2001)).
2. (**Single-input pole placement via linear systems**). Consider the following algorithm (Datta and Datta (1986)) for the single-input Hessenberg problem (H, \hat{b}) , where H is an unreduced upper Hessenberg matrix and $\hat{b} = (\alpha, 0, \dots, 0)^T, \alpha \neq 0$. Let $\{\mu_i\}_{i=1}^n$ be the eigenvalues to be assigned.

Step 1. Solve the $n \times n$ Hessenberg systems:

$$(H - \mu_i I) t_i = \hat{b}, i = 1, 2, \dots, n$$

Step 2. Solve for d :

$$T^T d = r,$$

where $T = (t_1, t_2, \dots, t_n)$ and $r = (\alpha, \alpha, \dots, \alpha)^T$.

Step 3. Compute $f^T = \frac{1}{\alpha} d^T$.

- (a) Give a proof of this algorithm, that is, prove that $\Omega(H - \hat{b} f^T) = \{\mu_1, \dots, \mu_n\}$; making necessary assumptions.
(Hint: Take $\Lambda = \text{diag}(\mu_1, \mu_2, \dots, \mu_n)$ in the proof of Algorithm 11.2.1 and follow the lines of the proof there.).
- (b) Prove that T in Step 2 is nonsingular if the entries in the set $\{\mu_1, \mu_2, \dots, \mu_n\}$ are pairwise distinct and none of them is an eigenvalue of H .
3. Consider the following modification of Algorithm in Exercise 2, proposed by Bru, Mas, and Urbano (1994a):

Step 1. For $i = 1, 2, \dots, n$ do

If μ_i is not in the spectrum of H , then solve the system

$$(H - \mu_i I) t_i = \hat{b}$$

Else solve the system $(H - \mu_i I) t_i = 0$.

Step 2. Define the vector $u = (u_1, \dots, u_n)^T$ as follows:

$u_i = 1$, if μ_i is an eigenvalue of H

$u_i = 0$, otherwise.

Step 3. Solve for f :

$$f^T T = u^T,$$

where $T = (t_1, t_2, \dots, t_n)$.

- (a) Give a proof of this algorithm assuming that the pair (H, \hat{b}) is controllable and that the numbers in the set $\{\mu_1, \mu_2, \dots, \mu_n\}$ are closed under complex conjugation and pairwise distinct.
- (b) Give an example to show that the assumption that $\mu_i, i = 1, \dots, n$ are pairwise distinct, cannot be relaxed.

(Note: Bru, Mas, and Urbana(1994) have given a more general algorithm which can assign multiple eigenvalues (Algorithm III in that paper)).

4. (Assigning Canonical Forms, Datta and Datta (1990))

Extend Algorithm 11.2.1 to the problems of assigning the following canonical forms: a companion matrix, an unreduced upper Hessenberg matrix, a Jordan matrix with no two Jordan blocks having the same eigenvalue. (**Hint:** Follow the line of the development of Algorithm 11.2.1 replacing Λ by the appropriate canonical form to be assigned).

5. (Multi-input Pole-placement via Linear Systems (Datta (1989))

Develop a multi-input version of the Algorithm in Exercise 2, making necessary assumptions.

Hint: Let $(H, \tilde{B} = \begin{pmatrix} R \\ O \end{pmatrix})$ be the controller Hessenberg pair.

Partition the set $\{\mu_1, \dots, \mu_n\}$ into $\left[\mu_1^{(1)}, \mu_1^{(2)}, \dots, \mu_1^{(n_1)}; \mu_2^{(1)}, \dots, \mu_2^{(n_2)}; \dots \mu_k^{(1)}, \dots, \mu_k^{(n_k)} \right]$, where $n_i, i = 1, \dots, k$ is the dimension of the i th subdiagonal block of H , and then solve $HL_i - L_i\Lambda_{ii} = \begin{pmatrix} R^{(i)} \\ O \end{pmatrix}$, where $R^{(i)}$ is the matrix of the first n_i columns of R .

(Note. Similar algorithms with different partitioning of the spectrum have appeared in Bru, Cerdan, and Urbano (1994b), and in Tusi (1986)).

6. Give a proof of Algorithm 11.2.3 (The RQ Formulation of Algorithm 11.2.1)). Consult Arnold and Datta (1998), if necessary.

7. Show that that explicit formula for the single-input pole assignment problem (Formula 11.2.7) is a Hessenberg-form of the Ackermann formula.

8. (Eigenvector Method for the Pole-Placement (Petkov, Christov, and Konstantinov (1984)).

- (a) Given the single-input controller-Hessenberg pair (H, \hat{b}) , show that it is possible to find an eigenvector \tilde{v}_+ , corresponding to an eigenvalue μ to be assigned, for the closed-loop matrix $H - bf^T$, without knowing the feedback vector f .
- (b) Let $\Lambda = \text{diag}(\mu_1, \mu_2, \dots, \mu_n)$ be the matrix of the eigenvalues to be assigned, and V be the eigenvector matrix and $v^{(1)}$ be the first row of V . Assume that $\mu_i, i = 1, \dots, n$

are all distinct. Prove that the feedback vector f can be computed from

$$f = \frac{1}{\alpha}(h_1 - v^{(1)}\Lambda V^{-1}),$$

where h_1 is the first row of H , and α is the 1st entry of \hat{b} .

- (c) What are the possible numerical difficulties of the above method for computing f ? Give an example to illustrate these difficulties.
 - (d) Following the same procedure as in the RQ formulation of **Algorithm 11.2.1**, work out an RQ version of the above eigenvector method. (Consult Arnold and Datta (1998), if necessary).
9. **An Explicit QR Algorithm** (Miminis and Paige (1982)). The explicit QR algorithm of Miminis and Paige finds an orthogonal matrix Q such that $Q^T(H - \beta e_1 f^T)Q = R$ is upper triangular with the desired eigenvalues on the diagonal. The algorithm has a forward sweep which determines Q and a backward sweep which computes f and R . Here is the RQ version of this algorithm (see Arnold and Datta (1998)).

Inputs: H —An $n \times n$ unreduced upper Hessenberg matrix; β , a nonzero scalar; and $S = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$.

Output: f —The feedback vector such that $\Omega(H - \beta e_1 f^T) = S$

Step 1. Set $H_1 = H$, and $\beta_1 = \beta$.

```

For  $i = 1, 2, \dots, n-1$  do
    Compute  $(H_i - \lambda_i I)Q_i^T = R_i$ , the RQ factorization of  $(H_i - \lambda_i I)$ 
    Compute  $\tau_i = r_{11}^{(i)} / \beta_i$  and  $\beta_{i+1} = q_{21}^{(i)} \beta_i$ 
    Compute  $H_{i+1}$ , where  $Q_i R_i + \lambda_i I = \begin{pmatrix} * & * \\ 0 & H_{i+1} \end{pmatrix}$ 
End

```

Step 2. Compute $f_n = (H_n - \lambda_n) / \beta_n$, and then f_{n-1} through f_1 as follows.

```

For  $i = n-1, n-2, \dots, 1$  do
    Compute  $f_i = Q_i^T \begin{pmatrix} \tau_i \\ f_{i+1} \end{pmatrix}$ 
End

```

Step 3. Set $f = f_1$.

- (a) Give a proof of the Algorithm.
- (b) Give a flop-count of the Algorithm.
- (c) Do Example 11.2.2 using the Algorithm.

10. **An Implicit QR Algorithm** (Patel and Misra (1984)). An implicit version of the Algorithm in Example 9 computes Q_i implicitly as the product $Q_i = U_i P_i$, where P_i is an orthogonal matrix such that $e_n^T (H_i - \lambda_i I) P_i^T = \alpha e_n^T$, and matrix U_i is an orthogonal matrix such that $U_i P_i H_i P_i^T U_i^T$ is another Hessenberg matrix; that is, the matrix U_i “**chases the bulge**” up the subdiagonal of $P_i H_i P_i^T$. This implicit version also incorporates the computation of the vector f in the forward sweep. The overall algorithm is described as follows:

Inputs: H —An unreduced $n \times n$ Hessenberg matrix, $\beta \neq 0, S = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$

Output: f —The feedback vector such that $\Omega(H - \beta e_1 f^T) = S$

Step 1. (**Initialization**) Set $H_n = H$, $\beta_n = \beta$, $Q = I_{n \times n}$, $f = 0_{n \times 1}$

Step 2. For $i = n, n-1, \dots, 2$ do

Step 2.1. Compute $P_i = \begin{pmatrix} I_{i-2} & 0 \\ 0 & \hat{P}_i \end{pmatrix}$, where \hat{P}_i is a Givens rotation such that $(h_{i,i-1}^{(i)}, h_{ii}^{(i)} - \lambda_i) \hat{P}_i^T = (0, \alpha)$, where $H_i = (h_{ij}^{(i)})$

Step 2.2. Chase up the bulge of the matrix $P_i H_i P_i^T$

For $j = i-1, i-2, \dots, 2$ do

Compute right Givens rotation $J_{j,j-1}^T$ that will eliminate the $(j+1, j-1)$ -th element of the matrix $J_{j+1,j} \cdots J_{i-1,i-2} P_i H_i P_i^T J_{i-1,i-2}^T \cdots J_{j+1,j}^T$ working on the j -th and $(j-1)$ -th columns

End

Set $U_i = J_{21} J_{32} \cdots J_{i-1,i-2} P_i = \begin{pmatrix} y_i^T \\ Q_i \end{pmatrix}$, $\hat{H}_i = U_i H_i U_i^T$

Step 2.3. Set $\beta_{i-1} = u_{21}^{(i)} \beta_i$, $\tau_i = \hat{h}_{21}^{(i)} / \beta_{i-1}$, where $\hat{H}_i = (\hat{h}_{jk}^{(i)})$.

Step 2.4. Update $f \equiv f + \tau_i Q^T y_i^T$, $Q \equiv Q_i Q$

Step 2.5. Extract H_{i-1} such that $\hat{H}_i = \left(\begin{array}{c|c} * & * \\ \hline 0 & H_{i-1} \end{array} \right)$

End

Step 3. Update $f \equiv f + \frac{(h_{11}^{(1)} - \lambda_1)}{\beta_1} Q$

(a) Give a proof of the algorithm and explore the differences of the algorithm with its explicit version derived in Example 9 (consult Arnold and Datta (1998)).

(b) Give a flop-count of the algorithm.

(c) Do Example 11.2.2 using the algorithm.

11. Explain the relationships between the RQ versions of the explicit QR, implicit QR, and the eigenvector methods for the single-input problem. Consult Arnold and Datta (1998), if necessary.
12. Modify the Schur algorithm (**Algorithm 11.3.3**) for the multi-input problem to handle the case when the complex numbers in the matrix Γ are not so ordered that the ordering

agrees with the diagonal structure of the matrix A_2 . Work out a simple example with this modified Schur method.

13. Modify the multi-input explicit QR algorithm in Section 11.3.2 so that it can be implemented without using complex arithmetic.
14. (**Eigenvalue Assignment via Sylvester Matrix Equation**). The following algorithm by Bhattacharyya and DeSouza (1982) solves the multi-input eigenvalue assignment problem:

1. Pick a matrix G arbitrarily
2. Solve the Sylvester equation $AX - X\tilde{A} = -BG$, where \tilde{A} is a matrix containing the eigenvalues $\{\lambda_1, \dots, \lambda_n\}$ to be assigned, for a full-rank solution X .
If the solution matrix X does not have full-rank, return to Step 1 and pick another G .
3. Compute the feedback matrix F by solving $FX = G$.

Give a proof of the algorithm and construct an example to illustrate the algorithm. (For the conditions on the existence of full rank solution of the Sylvester equation, see Chapter 12 and the paper by DeSouza and Bhattacharyya (1981)).

15. Construct an example to demonstrate that even if the feedback (vector) for the single-input problem is computed reasonably accurately, the closed-loop eigenvalues may still differ from those to be assigned.
16. (**Sensitivity Analysis of the Single-input Pole-placement Problem via Cauchy Matrix**; Mehmann and Xu (1998), Calvetti, Lewis and Reichel (1999)). Let $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ with $\lambda_i \neq \lambda_j$ for $i \neq j$. Define $e = (1, 1, \dots, 1)^T$. Let $S = \{\mu_1, \mu_2, \dots, \mu_n\}$, the eigenvalue set to be assigned; μ_i 's are distinct and none of them is in the spectrum of Λ .
 - (a) Prove that the vector f_e defined by
$$f_e = C_h^{-T}e$$

is such that $\Omega(\Lambda - ef_e^T) = S$, where $C_h = (c_{ij})$ is the Cauchy matrix:

$$c_{ij} = \frac{1}{\lambda_i - \mu_j}.$$
 - (b) Show that $\text{Cond}_2(C_h)$ is the spectral condition number of the closed-loop matrix $\Lambda - ef_e^T$.
 - (c) Using the result in (a) find an expression for the feedback vector f such that $\Omega(A - bf^T) = S$, assuming that A is diagonalizable.

- (d) Give a bound of the condition number of the eigenvector matrix of the closed-loop matrix $A - bf^T$ in terms of the condition number of C_h , the condition number of the eigenvector matrix X of A , and the minimum and maximum entries of the vector $X^{-1}b$.
- (e) Give a bound of the feedback vector f in terms of the norm of f_e and the norm of the inverse of the matrix XR , where $R = \text{diag}(\hat{b}_1, \hat{b}_2, \dots, \hat{b}_n)^T$, and $X^{-1}b = (\hat{b}_1, \hat{b}_2, \dots, \hat{b}_n)^T$.
- (f) From the bounds obtained in (d) and (e), verify the validity of some of the factors responsible for the ill-conditioning of the single-input eigenvalue assignment problem, established in Section 11.5.
17. From the expression (11.6.4) of the feedback matrix K , prove that if the condition of the eigenvector matrix is minimized, then a bound of the norm of the feedback matrix K is also minimized.
- Give an example to show that this does not necessarily mean that the feedback matrix will have the minimal norm.
18. Give an implementation of the robust eigenvalue assignment Algorithm (Algorithm 11.6.1) using the controller-Hessenberg form of the pair (A, B) . Work out an example to illustrate the algorithm.
19. (**Kautsky, Nichols, and Van Dooren (1985)**). Give a proof of Theorem 11.6.3; that is, prove the following results:
- (a) If the feedback matrix K is such that $(A - BK)$ has only stable eigenvalues λ_j , then the perturbed closed-loop matrix $A - BK + \Delta$ remains stable for all perturbations Δ that satisfy
- $$\|\Delta\|_2 \leq \min_{s=j\omega} \sigma_n\{sI - (A - BK)\} = \delta(K)$$
- (b)
- $$\delta(K) \geq \min_j \text{Re}(-\lambda_j)/\text{Cond}_2(X),$$
- where X is the eigenvector matrix.
- (c) Prove the results analogous to 19(a) and 19(b) in the discrete-time case.
20. (**Deadbeat-Control**). Given the discrete-system:

$$x_{i+1} = Ax_i + Bu_i,$$

the problem of “deadbeat” control is the problem of finding a state-feedback $u_i = -Kx_i + v_i$ such that the resulting system:

$$x_{i+1} = (A - BK)x_i + v_i$$

has the property that $(A - BK)^p = 0$ for some $p < n$ and $\text{rank}(B) > 1$.

The solution of the homogeneous part of the closed-loop system then “dies out” after p steps; and that is why the name **deadbeat control**.

A numerically reliable algorithm for the deadbeat control has been provided by Van Dooren (1984). The basic idea of the algorithm is as follows:

If $H = (H_{ij})$, $\tilde{B} = \begin{pmatrix} B_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ is the controller-Hessenberg pair of (A, B) , then the solution of the problem is equivalent to finding a feedback matrix K such that

$$V^T(H - \hat{B}K)V = \begin{pmatrix} 0 & H_{12} & \cdots & \cdots & \cdots & H_{1p} \\ 0 & & H_{23} & \cdots & \cdots & H_{2p} \\ \vdots & \ddots & & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & H_{p-1,p} \\ 0 & & & & & 0 \end{pmatrix},$$

for some orthogonal matrix V . The form on the right hand side is called the **“deadbeat” form**.

Note that in this case $(H - \hat{B}K)^p = 0$. Van Dooren’s algorithm finds K recursively in p steps. At the end of the i th step, one obtains the matrices V_i^T and K_i such that

$$V_i^T(H - \hat{B}K_i)V_i = \begin{pmatrix} H_d^i & * \\ 0 & H_h^i \end{pmatrix},$$

where the matrix H_d^i is in “deadbeat” form again, and the pair $(H_h^i, \hat{B}_h^i); V_i^T \hat{B} = \begin{pmatrix} \hat{B}_d^i \\ \hat{B}_h^i \end{pmatrix}$, is still in block-Hessenberg form.

Develop a scheme for obtaining V_i and hence complete the algorithm for “deadbeat” control problem. (Consult Van Dooren’s paper as necessary).

Work out an illustrative example.

Research Problems on Chapter 11

1. Carry out a round-off error analysis of the implicit QR algorithm of Patel and Misra (**Algorithm 11.2.5**) to establish the numerical stability of the algorithm.
2. Carry out a round-off error analysis of the recursive multi-input algorithm of Arnold and Datta (1990) (**Algorithm 11.3.1**). The algorithm is believed to be reliable in practice. Prove or disprove this using the results of your analysis.

3. An explicit expression for the family of feedback matrices for the multi-input eigenvalue assignment problem has been given in Arnold (1993). Use this expression to establish the fact that the sensitivities of the closed-loop eigenvalues depend upon the nearness of the system to an uncontrollable system, the separation of the open-loop and the closed-loop eigenvalues and, the ill-conditioning of the closed-loop eigenvector matrix.
4. Work out a RQ version of the recursive algorithm for the multi-input EVA problem by Arnold and Datta (1990) (**Algorithm 11.3.1**).
5. Carry out a round-off error analysis of the Schur algorithm of Varga (**Algorithm 11.3.3**) to establish the numerical stability of the algorithm.
6. In the QR algorithm of Miminis and Paige (1988) for the multi-input eigenvalue assignment problem, explicit shifting is used for the allocation of each eigenvalue. Work out an **implicit version** of this algorithm.

References

1. J. Ackermann, Der entwurf linear regelungssysteme im zustandsraum. *Regelungstechnik und prozessdatenverarbeitung*, vol. 7, 297-300, 1972
2. M. Arnold, *Algorithms and Conditioning for Eigenvalue Assignment*, Ph.D. Dissertation, Northern Illinois University, DeKalb, May 1993.
3. M. Arnold and B.N. Datta, An algorithm for the multi input eigenvalue assignment problem, *IEEE Trans. Automat. Contr.*, vol. 35, no. 10, pp. 1149-1152, 1990.
4. M. Arnold and B.N. Datta, The Single-input eigenvalue assignment algorithms: A close-look, *SIAM J. Matrix Anal. Appl.*, vol. 19, no. 2, pp. 444-467, 1998.
5. D. Bakshi, K. B. Datta, and G. D. Roy, Parallel algorithms for pole assignment of multi-input systems, *IEEE Proc. Control Theory Appl.*, vol. 141, no. 6, pp.367-372, 1994.
6. S.P. Bhattacharyya and E. DeSouza, Pole assignment via Sylvester's equation, *Syst. Contr. Letter.*, vol. 1, pp. 261-283, 1982.
7. R. Bru, J. Mas and A. Urbano, An Algorithm for the single input pole assignment problem, *SIAM J. Matrix Anal. Appl.*, pp. 393-407, 1994a.
8. R. Bru, J. Cerdan, and A. Urbano, An algorithm for the multi-input pole assignment problem, *Lin. Alg. Appl.*, vol. 1999, pp. 427-444, 1994b.
9. R. Bru, J. Cerdan, P. Fernandez de Cordoba, and A. Urbano, *A parallel algorithm for the partial single-input pole assignment problem*, *Appl. Math. Letters*, vol. 7, pp. 7-11, 1994c.

10. R. Byers and S. G. Nash, *Approaches to robust pole assignment*. Int. J. Control, vol. 49, no. 1, pp. 97-117, 1989
11. D. Calvetti, B. Lewis, and L. Reichel, on the Selection of poles in the Single-input pole placement problem, *Lin. Alg. Appl.*, vols. 302-303, pp. 331-345, 1999.
12. J. Carvalho and B.N. Datta, A block algorithm for the multi-input eigenvalue assignment problem, *Proc. 1st IFAC/IEEE Symposium on Systems, Structure and Control*, Prague, 2001.
13. R. K. Cavin and S. P. Bhattacharyya, Robust and well conditioned eigenstructure assignment via Sylvester's equation, *Optim. Control Appl. Methods*, vol. 4, pp.205-212, 1983.
14. C.-T. Chen, *Linear System Theory and Design*, CBS College Publishing, New York, 1984
15. E.K. Chu, N.K. Nichols and J. Kautsky, Robust pole assignment for output feedback, *Proc. 4th IMA Conference on Control Theory*, 1984.
16. K. -W. E. Chu, A controllability condensed form and a state feedback pole assignment algorithm for descriptor systems, *IEEE Trans. Automat., Control*, vol. 33, pp. 366-370, 1988.
17. M. G. Coutinho, A. Bhaya, and B. N. Datta, Parallel algorithms for the eigenvalue assignment problem in linear systems, *Proc. Int. Conference on Control and Information, Hong Kong*, pp.163-168, 1995.
18. C.L. Cox and W.F. Moss, Backward error analysis for a pole assignment algorithm, *SIAM J. Matrix Anal. Appl.*, vol. 10, no. 4, pp. 446-456, 1989.
19. C.L. Cox and W.F. Moss, Backward error analysis of a pole assignment algorithm II: The Complex Case, *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 1159-1171, 1992
20. B.N. Datta, Parallel and large-scale matrix computations in control: Some ideas, *Lin. Alg. Appl.*, vol. 121, pp. 243-264, 1989.
21. B.N. Datta and K. Datta, Efficient parallel algorithms for controllability and eigenvalue assignment problems, Proc. 25th IEEE Conf. Dec. Control, Athens, Greece, pp. 1611-1616, 1986.
22. B.N. Datta, An algorithm to assign eigenvalues in a Hessenberg matrix: single input case, *IEEE Trans. Automat. Contr.*, vol. AC-32, no. 5, pp. 414-417, 1987
23. B.N. Datta and K. Datta, On eigenvalue and canonical form assignments. *Lin. Alg. Appl.*, vol.131, pp. 161-182, 1990.

24. B.N. Datta, Parallel algorithms in control theory, *Proc. IEEE Conf. on Dec. Control*, pp. 1700-1704, 1991.
25. B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, Pacific Grove, CA, 1995.
26. B.N. Datta, S. Elhay, and Y. Ram, Orthogonality and partial pole assignment for the symmetric definite pencil, *Lin. Alg. Appl.*, vol. 257, pp. 29-48, 1997.
27. B.N. Datta and Y. Saad, Arnoldi methods for large Sylvester-like matrix equations and an associated algorithm for partial spectrum assignment, *Lin. Alg. Appl.*, vol. 154-156, pp.225-244, 1991.
28. K. Datta, The matrix equation $XA - BX = R$ and its applications, *Lin. Alg. Appl.*, vol. 109, pp. 91-105, 1988.
29. E. DeSouza and S.P. Bhattacharyya, Controllability, observability and the solution of $AX - XB = C$, *Lin. Alg. Appl.*, vol. 39, pp. 167-188, 1981.
30. L. R. Fletcher, J. Kautsky, and N. K. Nichols, Eigenstructure assignment in descriptor systems, *IEEE Trans. Automat. Control*, vol. AC-31, pp. 1138-1141, 1986.
31. C. He, A. J. Laub, and V. Mehrmann, *Placing plenty of poles is pretty preposterous*, *DFG-Forschergruppe Scientific Parallel Computing, Preprint 95-17*, Fak. f. Mathematik, TU Chemnitz-Zwickau, D-09107, Chemnitz, FRG, 1995.
32. N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.
33. T. Kailath, *Linear Systems*. Prentice Hall, Englewood Cliffs, NJ, 1980.
34. J. Kautsky, N. K. Nichols, and K. -W. E. Chu, Robust pole assignment in singular control systems, *Lin. Alg. Appl.*, vol. 121, pp. 9-37, 1989.
35. J. Kautsky, N.K. Nichols and P. Van Dooren, Robust pole assignment in linear state feedback, *Int. J. Control*, vol. 41, no. 5, pp. 1129-1155, 1985.
36. L.H. Keel, Fleming J.A., and Bhattacharyya S.P., Pole assignment via Sylvester's equation, *Contemporary Mathematics, Amer. Math. Soc., Providence, RI*, vol. 47, pp. 265, 1985.(R. Brualdi, et al., Editors.)
37. M.M. Konstantinov and P. Petkov, Conditioning of linear state feedback, *Technical Report: 93-61*, Dept. of Engineering, Leicester University, 1993.
38. A.J. Laub and A. Linnemann, Hessenberg forms in linear systems theory, *Computational and Combinatorial Methods in Systems Theory*, Elsevier Science publishers, North-Holland, pp. 229-244, 1986, (C.I. Byrnes and A. Lindquist, Editors).

39. MathWorks, Inc., *The MATLAB User's Guide*, The MathWorks, Inc., Natick, MA, 1992.
40. V. Mehrmann and H. Xu, An analysis of the pole placement problem I: The single-input case, *Electron. Trans. Numer. Anal.*, vol.4, pp. 89-105, 1996.
41. V. Mehrmann and H. Xu, An analysis of the pole placement problem II: The multi-input Case, *Electron. Trans. Numer. Anal.*, vol. 5, pp. 77-97, 1997.
42. V. Mehrmann and H. Xu, Choosing the poles so that the single-input pole placement problem is well-conditioned, *SIAM J. Matrix Anal. Appl.*, 1998.
43. G.S. Miminis and C.C. Paige, An algorithm for pole assignment of time-invariant linear systems. *Int. J. Control*, vol. 35, no. 2, pp. 341-354, 1982.
44. G.S. Miminis, *Numerical Algorithms for Controllability and Eigenvalue Allocation*. M.Sc. Thesis, School of Computer Science, McGill University, Montreal, Canada, 1981.
45. G.S. Miminis and C.C. Paige, A direct algorithm for pole assignment of time-invariant multi-input linear systems using state feedback, *Automatica*, vol. 24, no. 3, pp. 343-356, 1988.
46. G.S. Miminis, *Polepack*, A collection of MATLAB programs for eigenvalue assignment, available on *NETLIB* (www.netlib.org), 1991.
47. P. Misra and R.V. Patel, Numerical algorithms for eigenvalue assignment by constant and dynamic output feedback, *IEEE Trans. Automat. Contr.*, vol. 34, no. 6, pp. 579-580, 1989.
48. R.V. Patel and P. Misra, Numerical algorithms for eigenvalue assignment by state feedback, *Proc. IEEE*, vol. 72, no. 12, pp. 1755-1764, 1984.
49. P. Petkov, N.D. Christov and M.M. Konstantinov, A computational algorithm for pole assignment of linear multi input systems, *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 11, pp. 1044-1047, 1986.
50. P. Petkov, N.D. Christov and M.M. Konstantinov, A computational algorithm for pole assignment of linear single-input systems, *IEEE Trans. Automat. Contr.*, vol. AC-29, no. 11, pp. 1045-1048, 1984.
51. J. Rice, Theory of Conditioning, *SIAM J. Numer. Anal.*, vol. 3, no.2, pp. 287-311, 1966.
52. Y. Saad, Projection and deflation methods for partial pole assignment in linear state feedback, *IEEE Trans. Automat. Control*, vol. 33, pp. 290-297, 1988.
53. B. Shafai and S. P. Bhattacharyya, An algorithm for pole placement in high-order multi-variable systems, *IEEE Trans. Automat. Control*, vol. 33, 9, pp. 870-876, 1988.

54. G.W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.
55. J.-G. Sun, Perturbation analysis of the pole assignment problem, *SIAM J. Matrix Anal. Appl.*, vol. 17, pp. 313-331, 1996.
56. F. Szidarovszky and A.T. Bahill, *Linear Systems Theory*, CRC Press, Boca Raton, 1991.
57. A.L. Tits and Y. Yang, Globally convergent algorithms for robust pole assignment by state feedback, *IEEE Trans. Automat. Control*, AC-41, pp. 1432-1452, 1996.
58. C.C. Tsui, An algorithm for computing state feedback in multi input linear systems, *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 3, pp. 243-246, 1986.
59. M. Valasek and N. Olgac, Efficient eigenvalue assignments for general linear MIMO systems, *Automatica*, vol. 31, pp. 1605-1617, 1995.
60. M. Valasek and N. Olgac, Efficient pole placement technique for linear time-variant SISO systems, *Proc. IEEE Control Theory Appl.*, vol. 142, 451-458, 1995.
61. P. Van Dooren, Deadbeat Control: A special inverse eigenvalue problem, *BIT*, vol.24, pp. 681-699, 1984.
62. P.M. Van Dooren and M. Verhaegen, On the use of unitary state-space transformations, *Contemporary Math.*, Amer. Math. Soc., Providence, RI, vol. 47, pp. 447-463, 1985. (R. Brualdi, et al. Editors).
63. A. Varga, A multishift Hessenberg method for pole assignment of single-input systems, *IEEE Trans. Automat. Control*, vol. 41, pp. 1795-1799, 1996.
64. A. Varga, Robust pole assignment for descriptor systems, *Proc. Mathematical Theory of Networks and Systems* (MTNS '2000), 2000.
65. A. Varga, A Schur method for pole assignment, *IEEE Trans. Automat. Contr.*, vol. AC-26, no. 2, pp. 517-519, 1981.
66. S.-F. Xu, *An introduction to inverse algebraic eigenvalue problems*, Peking University Press, Peking, China, 1998.
67. J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendem Press, Oxford, England, 1965.

Chapter 12

STATE ESTIMATION: Observer and the Kalman Filter

Contents

12.1	Introduction	530
12.2	State Estimation via Eigenvalue Assignment	531
12.3	State Estimation via Sylvester Equation	532
12.4	Reduced-order State Estimation	535
12.4.1	Reduced-order State Estimation via Eigenvalue Assignment	535
12.4.2	Reduced-order State Estimation via Sylvester-observer Equation	540
12.5	Combined State Feedback and Observer Design	542
12.6	Characterization of Nonsingular Solutions of the Sylvester Equation	544
12.7	Numerical Solutions of the Sylvester-Observer Equation	546
12.7.1	A Recursive Method for the Hessenberg Sylvester-Observer Equation.	547
12.7.2	A Recursive Block-Triangular Algorithm for the Hessenberg Sylvester-Observer Equation	551
12.8	Numerical Solution of a Constrained Sylvester-observer Equation	556
12.9	Optimal State Estimation: The Kalman Filter	560
12.10	The Linear Quadratic Gaussian Problem	565
12.11	Some Selected Software	570
12.11.1	MATLAB CONTROL SYSTEM TOOLBOX	570
12.11.2	MATCONTROL	570
12.11.3	CSP-ANM	570
12.11.4	SLICOT	570
12.11.5	MATRIX _X	570
12.12	Summary and Review	571
12.13	Chapter Notes and Further Reading	574

Topics Covered

- State Estimation via Eigenvalue Assignment
- State Estimation via Sylvester-observer Equation
- Characterization of the Unique Nonsingular Solution to the Sylvester Equation
- Numerical Methods for the Sylvester-observer Equation.
- A Numerical Method for the Constrained Sylvester-observer Equation
- Kalman Filter
- Linear Quadratic Gaussian (LQG) Design.

12.1 Introduction

We have seen in Chapter 10 that all the state-feedback problems, such as feedback stabilization, eigenvalue and eigenstructure assignment, the LQR and the state-feedback H_∞ control problems, etc., require that the state vector $x(t)$ should be explicitly available. However, in most practical situations, the states are not fully accessible and all the designer knows are the output $y(t)$ and the input $u(t)$. The unavailable states, somehow, need to be estimated accurately from the knowledge of the matrices A , B , and C , the output vector $y(t)$, and the input vector $u(t)$. In this chapter, we discuss how the states of a continuous-time system can be estimated. **The discussions here apply equally to the discrete-time systems, possibly with some minor changes.** So we concentrate on the continuous-time case only.

We describe two common procedures for state estimation: **one, via eigenvalue assignment and the other, via solution of the Sylvester-observer equation.**

The Hessenberg-Schur method for the Sylvester equation, described in Chapter 8, can be used for numerical solution of the Sylvester-observer equation. We, however, describe two other numerical methods (**Algorithms 12.7.1** and **12.7.2**), especially designed for this equation. Both are based on the reduction of the observable pair (A, C) to the observer-Hessenberg pair, described in Chapter 6 and are recursive in nature. *Algorithm 12.7.2 is a block-generalization of Algorithm 12.7.1 and seems to be a little more efficient than the later.* Algorithm 12.7.2 is also suitable for high performance computing. Both seem to have good numerical properties.

The chapter concludes with a well-known procedure due to Kalman (**Kalman filtering**) for optimal estimation of the states of a **stochastic system**, followed by a brief discussion on the Linear Quadratic Gaussian (**LQG**) problem that deals with optimization of a performance measure for a stochastic system.

12.2 State Estimation via Eigenvalue Assignment

Consider the linear time-invariant continuous-time system:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t),\end{aligned}\tag{12.2.1}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $C \in \mathbb{R}^{r \times n}$.

Let $\hat{x}(t)$ be an estimate of the state vector $x(t)$. Obviously, we would like to construct the vector $\hat{x}(t)$ in such a way that the error $e(t) = x(t) - \hat{x}(t)$ approaches zero as fast as possible. Suppose, we design a dynamical system using our available resources: the output variable $y(t)$, input variable $u(t)$, and the matrices A, B, C , satisfying

$$\dot{\hat{x}}(t) = (A - KC)\hat{x}(t) + Ky(t) + Bu(t),\tag{12.2.2}$$

where the matrix K is to be constructed. Then

$$\begin{aligned}\dot{e}(t) &= \dot{x}(t) - \dot{\hat{x}}(t) = Ax(t) + Bu(t) - A\hat{x}(t) + KC\hat{x}(t) - Ky(t) - Bu(t) \\ &= (A - KC)x(t) - (A - KC)\hat{x}(t) = (A - KC)e(t)\end{aligned}$$

The solution of this system of differential equations is $e(t) = e^{(A-KC)t}e(0)$, which shows that the rate at which the entries of the error vector $e(t)$ approach zero can be controlled by the eigenvalues of the matrix $A - KC$. For example, if all the eigenvalues of $A - KC$ have negative real parts less than $-\alpha$, then the error $e(t)$ will approach zero faster than $e^{-\alpha t}e(0)$.

The above discussion shows that the problem of state estimation can be solved by finding a matrix K such that the matrix $A - KC$ has a suitable desired spectrum. Note that if (A, C) is observable, then such K always exists; because, the observability of (A, C) implies the controllability of (A^T, C^T) . Also, if (A^T, C^T) is controllable, then by the Eigenvalue Assignment Theorem (Theorem 10.4.1), there always exists a matrix L such that $(A^T + C^T L)$ has an arbitrary spectrum. We can therefore choose $K = -L^T$ so that the eigenvalues of $A^T - C^T K^T$ (which are the same as those of $A - KC$) will be arbitrarily assigned.

Theorem 12.2.1 *If (A, C) is observable, then the states $x(t)$ of the system*

$$\dot{x}(t) = Ax(t) + Bu(t)\tag{12.2.3}$$

$$y(t) = Cx(t)\tag{12.2.4}$$

can be estimated by

$$\dot{\hat{x}}(t) = (A - KC)\hat{x}(t) + Ky(t) + Bu(t),\tag{12.2.5}$$

where K is constructed such that $A - KC$ is a stable matrix. The error $e(t) = x(t) - \hat{x}(t)$ is governed by

$$\dot{e}(t) = (A - KC)e(t)$$

and $e(t) \rightarrow 0$ as $t \rightarrow \infty$.

12.3 State Estimation via Sylvester Equation

We now present another approach for state estimation. Knowing $A, B, C, u(t)$ and $y(t)$, let's construct the system

$$\dot{z}(t) = Fz(t) + Gy(t) + Pu(t), \quad (12.3.1)$$

where F is $n \times n$, G is $n \times r$, and P is $n \times m$, in such a way that for some constant $n \times n$ nonsingular matrix X , the error vector $e(t) = z(t) - Xx(t) \rightarrow 0$ for any $x(0), z(0)$, and $u(t)$. The vector $z(t)$ will then be an estimate of $Xx(t)$. The system (12.3.1) is then said to be the **state observer** for the system (12.2.1). The idea originated with D. Luenberger (Luenberger (1964)) and is hence referred to in control theory as the **Luenberger observer**.

We now show that the system (12.3.1) will be a state observer if the matrices X, F, G and P satisfy certain requirements.

Theorem 12.3.1 (Observer Theorem) *The system (12.3.1) is a state-observer of the system (12.2.1); that is, $z(t)$ is an estimate of $Xx(t)$ in the sense that the error $e(t) = z(t) - Xx(t) \rightarrow 0$ as $t \rightarrow \infty$ for any initial conditions $x(0), z(0)$, and $u(t)$ if*

$$(1) \quad XA - FX = GC,$$

$$(2) \quad P = XB,$$

$$(3) \quad F \text{ is stable.}$$

Proof: We need to show that if the conditions (1)-(3) are satisfied, then $e(t) \rightarrow 0$ as $t \rightarrow 0$.

From $e(t) = z(t) - Xx(t)$, we have

$$\begin{aligned} \dot{e}(t) &= \dot{z}(t) - X\dot{x}(t) \\ &= Fz(t) + Gy(t) + Pu(t) - X(Ax(t) + Bu(t)). \end{aligned} \quad (12.3.2)$$

Substituting $y(t) = Cx(t)$ while adding and subtracting $FXx(t)$ in equation (12.3.2), we get

$$\dot{e}(t) = Fe(t) + (FX - XA + GC)x(t) + (P - XB)u(t).$$

If the conditions (1) and (2) are satisfied, then we obtain

$$\dot{e}(t) = Fe(t).$$

If, in addition, the condition (3) is satisfied, then clearly $e(t) \rightarrow 0$ as $t \rightarrow \infty$, for any $x(0), z(0)$ and $u(t)$.

Hence $z(t)$ is an estimate of $Xx(t)$. ■

The Sylvester-Observer Equation

Definition 12.3.1 *The matrix equation*

$$XA - FX = GC, \quad (12.3.3)$$

where A and C are given and X, F and G are to be found will be called the **Sylvester-observer equation**.

The name “**Sylvester-observer equation**” is justified, because the equation arises in construction of an observer and it is a variation of the classical Sylvester equation (discussed in Chapter 8):

$$XA + TX = R,$$

where A, T , and R are given and X is the only unknown matrix.

Theorem 12.3.1 suggests the following method for the observer design.

Algorithm 12.3.1 Full-order Observer Design via Sylvester-Observer Equation

Inputs: The system matrices A, B , and C of order $n \times n, n \times m$, and $r \times n$ respectively.

Output: An estimate $\hat{x}(t)$ of the state vector $x(t)$.

Assumptions: (A, C) is observable.

Step 1. Find a **nonsingular solution** X of the Sylvester-observer equation

$$XA - FX = GC$$

by choosing F as a stable matrix and choosing G in such a way that the resulting solution X is nonsingular.

Step 2. Compute $P = XB$.

Step 3. Construct the observer $z(t)$ by solving the system of differential equations:

$$\dot{z}(t) = Fz(t) + Gy(t) + Pu(t), \quad z(0) = z_0.$$

Step 4. Find an estimate $\hat{x}(t)$ of $x(t)$: $\hat{x}(t) = X^{-1}z(t)$.

Example 12.3.1

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \\ C = (1 \ 0).$$

(A, C) is observable.

Step 1. Choose $G = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$, $F = \text{diag}(-1, -3)$.

Then a solution X of $XA - FX = GC$ is

$$X = \begin{pmatrix} 0.6667 & -0.3333 \\ 0.8000 & -0.2000 \end{pmatrix}$$

(computed by MATLAB function **lyap**). The matrix X is nonsingular.

Step 2.

$$P = XB = \begin{pmatrix} 0.6667 \\ 0.8000 \end{pmatrix}.$$

Step 3. An estimate $\hat{x}(t)$ of $x(t)$ is

$$\hat{x}(t) = X^{-1}z(t) = \begin{pmatrix} -1.5 & 2.5 \\ -6 & 5 \end{pmatrix} \begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix} = \begin{pmatrix} -1.5z_1 + 2.5z_2 \\ -6z_1 + 5z_2 \end{pmatrix}$$

where $z(t) = \begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix}$ is given by

$$\dot{z}(t) = \begin{pmatrix} -1 & 0 \\ 0 & -3 \end{pmatrix} z(t) + \begin{pmatrix} 1 \\ 3 \end{pmatrix} y(t) + \begin{pmatrix} 0.6667 \\ 0.8000 \end{pmatrix} u(t).$$

Comparison of the State and Estimate for Example 12.3.1.

Below, we compare the estimate $\hat{x}(t)$, obtained by Algorithm 12.3.1, with the state $x(t)$, found by directly solving the equation (12.2.1) with $u(t)$ as the *unit step function*, and $x(0) = (6, 0)^T$. The differential equation in Step 3 was solved with $z(0) = 0$. The MATLAB function **ode23** was used to solve both the equations. The solid line corresponds to the exact states and the dotted line corresponds to the estimated state.

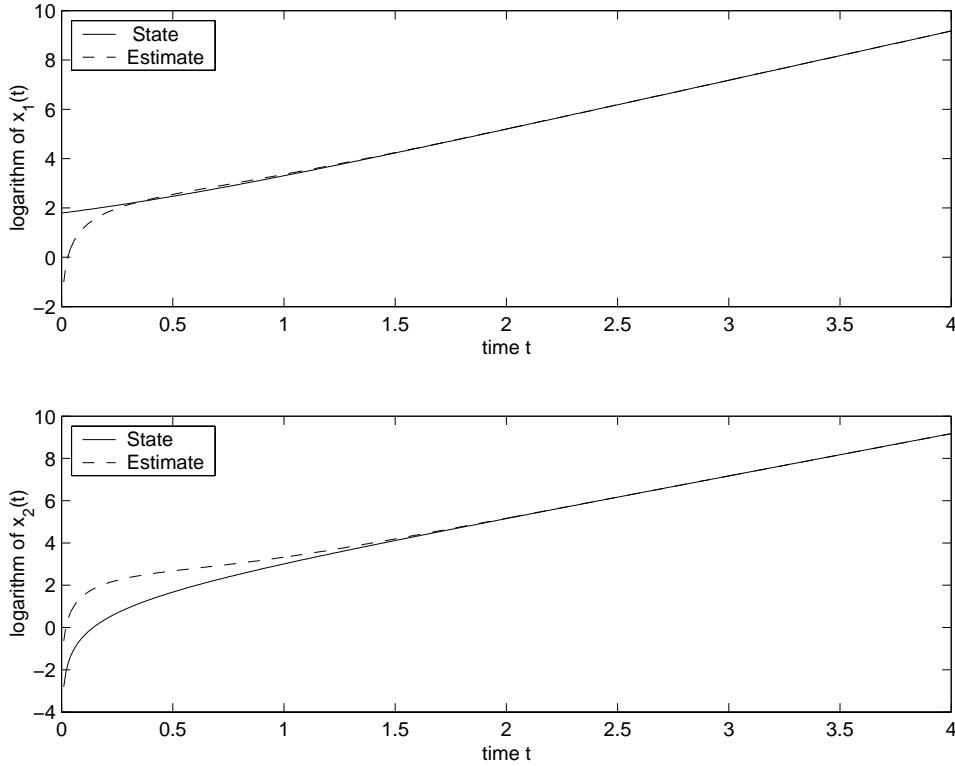


Figure 12.1: The First and Second Variables of the State $x(t)$ and Estimate $\hat{x}(t)$ for Example 12.3.1.

12.4 Reduced-order State Estimation

In this section, we show that if the $r \times n$ output matrix C has full rank r , then the problem of finding a full n th order state estimator for the system (12.2.1) can be reduced to the problem of finding an $(n - r)$ th order estimator.

Such an estimator is known as a **reduced-order estimator**. Once a reduced-order estimator of order $n - r$ rather than n is constructed, the full states of the original system can be obtained from the $(n - r)$ state variable of this observer together with the r variables available from measurements. As in the full-dimensional case, we will describe two approaches for finding a reduced-order estimator. We start with the **eigenvalue assignment approach**.

For the sake of convenience, in the next two sections, we will denote the vector $x(t)$ and its derivative $\dot{x}(t)$ just by x and \dot{x} . Similarly, for the other vectors.

12.4.1 Reduced-order State Estimation via Eigenvalue Assignment

Assume as usual that A is an $n \times n$ matrix, B is an $n \times m$ matrix ($m \leq n$), C is an $r \times n$ matrix with full rank ($r < n$), and (A, C) is observable.

Since C has full rank, we can choose an $(n - r) \times n$ matrix R such that the matrix $S = \begin{pmatrix} C \\ R \end{pmatrix}$ is nonsingular.

Introducing the new variable $\tilde{x} = Sx$, we can then transform the system (12.2.1) to

$$\begin{aligned}\dot{\tilde{x}} &= SAS^{-1}\tilde{x} + SBu \\ y &= CS^{-1}\tilde{x} = (I_r, 0)\tilde{x}.\end{aligned}\tag{12.4.1}$$

Let's now partition

$$\bar{A} = SAS^{-1} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \end{pmatrix}, \quad \bar{B} = SB = \begin{pmatrix} \bar{B}_1 \\ \bar{B}_2 \end{pmatrix},\tag{12.4.2}$$

and $\tilde{x} = \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix}$, where \bar{A}_{11} and \tilde{x}_1 are, respectively, $r \times r$ and $r \times 1$. Then we have

$$\begin{aligned}\begin{pmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_2 \end{pmatrix} &= \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix} + \begin{pmatrix} \bar{B}_1 \\ \bar{B}_2 \end{pmatrix} u \\ y &= (I_r, 0) \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix} = \tilde{x}_1.\end{aligned}$$

That is,

$$\dot{y} = \dot{\tilde{x}}_1 = \bar{A}_{11}\tilde{x}_1 + \bar{A}_{12}\tilde{x}_2 + \bar{B}_1 u\tag{12.4.3}$$

$$\dot{\tilde{x}}_2 = \bar{A}_{21}\tilde{x}_1 + \bar{A}_{22}\tilde{x}_2 + \bar{B}_2 u\tag{12.4.4}$$

$$y = \tilde{x}_1\tag{12.4.5}$$

Since $y = \tilde{x}_1$, we only need to find an estimator for the vector \tilde{x}_2 of this transformed system. The transformed system is not in standard state-space form. However, the system can be easily put in standard form by introducing the new variables

$$\bar{u} = \bar{A}_{21}\tilde{x}_1 + \bar{B}_2 u = \bar{A}_{21}y + \bar{B}_2 u,\tag{12.4.6}$$

and

$$v = \dot{y} - \bar{A}_{11}y - \bar{B}_1 u.\tag{12.4.7}$$

From (12.4.4)-(12.4.7), we then have

$$\begin{aligned}\dot{\tilde{x}}_2 &= \bar{A}_{22}\tilde{x}_2 + \bar{u} \\ v &= \bar{A}_{12}\tilde{x}_2\end{aligned}\tag{12.4.8}$$

which is in standard form.

Since (A, C) is observable, it can be shown [Exercise 3] that $(\bar{A}_{22}, \bar{A}_{12})$ is also observable. Since \tilde{x}_2 has $(n - r)$ elements, we have thus reduced the full n dimensional estimation problem to an $(n - r)$ dimensional problem. We, therefore, now concentrate on finding an estimate of \tilde{x}_2 .

By (12.2.2) an $(n - r)$ dimensional estimate $\hat{\tilde{x}}_2$ of \tilde{x}_2 defined by (12.4.8) is of the form

$$\dot{\hat{\tilde{x}}}_2 = (\bar{A}_{22} - L\bar{A}_{12})\hat{\tilde{x}}_2 + Lv + \bar{u},$$

for any matrix L chosen such that $\bar{A}_{22} - L\bar{A}_{12}$ is stable.

Substituting the expressions for \bar{u} and v from (12.4.6) and (12.4.7) into the last equation, we have

$$\begin{aligned}\dot{\hat{\tilde{x}}}_2 &= (\bar{A}_{22} - L\bar{A}_{12})\hat{\tilde{x}}_2 + L(\dot{y} - \bar{A}_{11}y - \bar{B}_1u) \\ &\quad + (\bar{A}_{21}y + \bar{B}_2u).\end{aligned}$$

Defining another new variable

$$z = \hat{\tilde{x}}_2 - Ly,$$

we can then write

$$\begin{aligned}\dot{z} &= (\bar{A}_{22} - L\bar{A}_{12})(z + Ly) + (\bar{A}_{21} - L\bar{A}_{11})y + (\bar{B}_2 - L\bar{B}_1)u \\ &= (\bar{A}_{22} - L\bar{A}_{12})z + [(\bar{A}_{22} - L\bar{A}_{12})L + (\bar{A}_{21} - L\bar{A}_{11})]y + (\bar{B}_2 - L\bar{B}_1)u\end{aligned}\tag{12.4.9}$$

Comparing the equation (12.4.9) with (12.2.2) and noting that $\bar{A}_{22} - L\bar{A}_{12}$ is a stable matrix, we see that $z + Ly$ is also an estimate of \tilde{x}_2 .

Once an estimate of \tilde{x}_2 is found, an estimate of the original n -dimensional state vector x from the estimate of \tilde{x}_2 can be easily constructed, as shown below.

Since $y = \tilde{x}_1$, and $\hat{\tilde{x}}_2 = z + Ly$, we immediately have

$$\hat{\tilde{x}} = \begin{pmatrix} \hat{\tilde{x}}_1 \\ \hat{\tilde{x}}_2 \end{pmatrix} = \begin{pmatrix} y \\ Ly + z \end{pmatrix}\tag{12.4.10}$$

as an estimate of \tilde{x} .

Finally, since $\tilde{x} = Sx$, an estimate \hat{x} of x can be constructed from an estimate of \tilde{x} as:

$$\hat{x} = S^{-1}\hat{\tilde{x}} = \begin{pmatrix} C \\ R \end{pmatrix}^{-1} \begin{pmatrix} y \\ Ly + z \end{pmatrix}.$$

The above discussion can be summarized in the following algorithm:

Algorithm 12.4.1 Reduced-order Observer Design via Eigenvalue Assignment

Inputs: The system matrices A, B, C , respectively, of order $n \times n, n \times m$ and $r \times n$.

Output: An estimate \hat{x} of the state vector x .

Assumptions: (1) (A, C) is observable. (2) C is of full rank.

Step 1. Find an $(n - r) \times n$ matrix R such that $S = \begin{pmatrix} C \\ R \end{pmatrix}$ is nonsingular.

Step 2. Compute $\bar{A} = SAS^{-1}$, $\bar{B} = SB$ and partition them as

$$\bar{A} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} \bar{B}_1 \\ \bar{B}_2 \end{pmatrix} \quad (12.4.11)$$

where $\bar{A}_{11}, \bar{A}_{12}, \bar{A}_{21}, \bar{A}_{22}$ are, respectively, $r \times r, r \times (n-r), (n-r) \times r$, and $(n-r) \times (n-r)$ matrices.

Step 3. Find a matrix L such that $\bar{A}_{22} - L\bar{A}_{12}$ is stable.

Step 4. Construct a reduced-order observer by solving the systems of differential equations:

$$\dot{z} = (\bar{A}_{22} - L\bar{A}_{12})z + [(\bar{A}_{22} - L\bar{A}_{12})L + (\bar{A}_{21} - L\bar{A}_{11})]y + (\bar{B}_2 - L\bar{B}_1)u, z(0) = z_0 \quad (12.4.12)$$

Step 5. Find \hat{x} , an estimate of x :

$$\hat{x} = \begin{pmatrix} C \\ R \end{pmatrix}^{-1} \begin{pmatrix} y \\ Ly + z \end{pmatrix}. \quad (12.4.13)$$

Example 12.4.1 Consider the design of a reduced-order observer for the Helicopter problem discussed in Doyle and Stein (1981), and also considered in Dorato et al (1995), with the following data:

$$A = \begin{pmatrix} -0.02 & 0.005 & 2.4 & -32 \\ -0.14 & 0.44 & -1.3 & -30 \\ 0 & 0.018 & -1.6 & 1.2 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0.14 & -0.12 \\ 0.36 & -8.6 \\ 0.35 & 0.009 \\ 0 & 0 \end{pmatrix}$$

and

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 57.3 \end{pmatrix}.$$

Since $\text{rank}(C) = 2$, $r = 2$.

Step 1. Choose $R = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$. The matrix $S = \begin{pmatrix} C \\ R \end{pmatrix}$ is nonsingular.

Step 2. $\bar{A} = SAS^{-1} = \begin{pmatrix} 1.7400 & -0.5009 & -0.1400 & -1.1600 \\ -57.3006 & -1 & 0 & 57.3000 \\ -0.0370 & -1.0698 & -0.1600 & 0.6600 \\ 2.3580 & -0.4695 & -0.1400 & -1.7600 \end{pmatrix}$,

$$\bar{B} = SB = \begin{pmatrix} 0.3600 & -8.6000 \\ 0 & 0 \\ 0.8500 & -8.7110 \\ 0.7100 & -8.5910 \end{pmatrix}.$$

$$\bar{A}_{11} = \begin{pmatrix} 1.74 & -0.5 \\ -57.3 & -1 \end{pmatrix}, \bar{A}_{12} = \begin{pmatrix} -0.14 & -1.16 \\ 0 & 57.3 \end{pmatrix}$$

$$\bar{A}_{21} = \begin{pmatrix} -0.0370 & -1.0698 \\ 2.3580 & -0.4695 \end{pmatrix}, \bar{A}_{22} = \begin{pmatrix} -0.1600 & 0.6600 \\ -0.1400 & -1.7600 \end{pmatrix}$$

$$\bar{B}_1 = \begin{pmatrix} 0.3600 & -8.6000 \\ 0 & 0 \end{pmatrix}, \bar{B}_2 = \begin{pmatrix} 0.85 & -8.711 \\ 0.7100 & -8.5910 \end{pmatrix}$$

Step 3. The matrix $L = \begin{pmatrix} -6 & -0.1099 \\ 1 & 0.0244 \end{pmatrix}$ is such that the eigenvalues of $\bar{A}_{22} - L\bar{A}_{12}$ (the observer eigenvalues) are $\{-1, -2\}$ (L is obtained using MATLAB function **place**).

Step 4. The reduced-order 2-dimensional observer is given by

$$\dot{z} = (\bar{A}_{22} - L\bar{A}_{12})z + [(\bar{A}_{22} - L\bar{A}_{12})L + (\bar{A}_{21} - L\bar{A}_{11})]y + (\bar{B}_2 - L\bar{B}_1)u$$

with $\bar{A}_{11}, \bar{A}_{12}, \bar{A}_{21}, \bar{A}_{22}, \bar{B}_1$, and \bar{B}_2 as computed above.

An estimate \hat{x} of the state vector x is then

$$\hat{x} = \begin{pmatrix} C \\ R \end{pmatrix}^{-1} \begin{pmatrix} y \\ Ly + z \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ -1 & -0.0175 & 0 & 1 \\ 0 & 0.0175 & 0 & 0 \end{pmatrix} \begin{pmatrix} y \\ Ly + z \end{pmatrix},$$

where z is determined from (12.4.13).

Remark: The explicit inversion of the matrix $S = \begin{pmatrix} C \\ R \end{pmatrix}$, which could be a source of large round-off errors in case this matrix is ill-conditioned, can be avoided by taking the QR decomposition of the matrix C : $C = RQ_1$ and then choosing an orthogonal matrix Q_2 such that the matrix $Q = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$ is orthogonal. The matrix Q then can be used in place of S . We leave the details for the readers as an exercise [Exercise 18].

Comparison of the State and Estimate for Example 12.4.1.

Below, we compare the estimate $\hat{x}(t)$, obtained by Algorithm 12.4.1 with the state $x(t)$, found by directly solving the equation (12.2.1) with $u(t) = H(t)[1 \ 1]^T$, $H(t)$ is the unit step function and $x(0) = (6, 0, 0, 0)^T$. To solve the equations (12.2.1) and (12.4.12), **MATLAB** function **ode23** was used. For equation (12.4.12), the initial condition was $z(0) = 0$. The first and the third components of the solutions are compared. The solid line corresponds to the exact state and the dotted line corresponds to the estimated state.

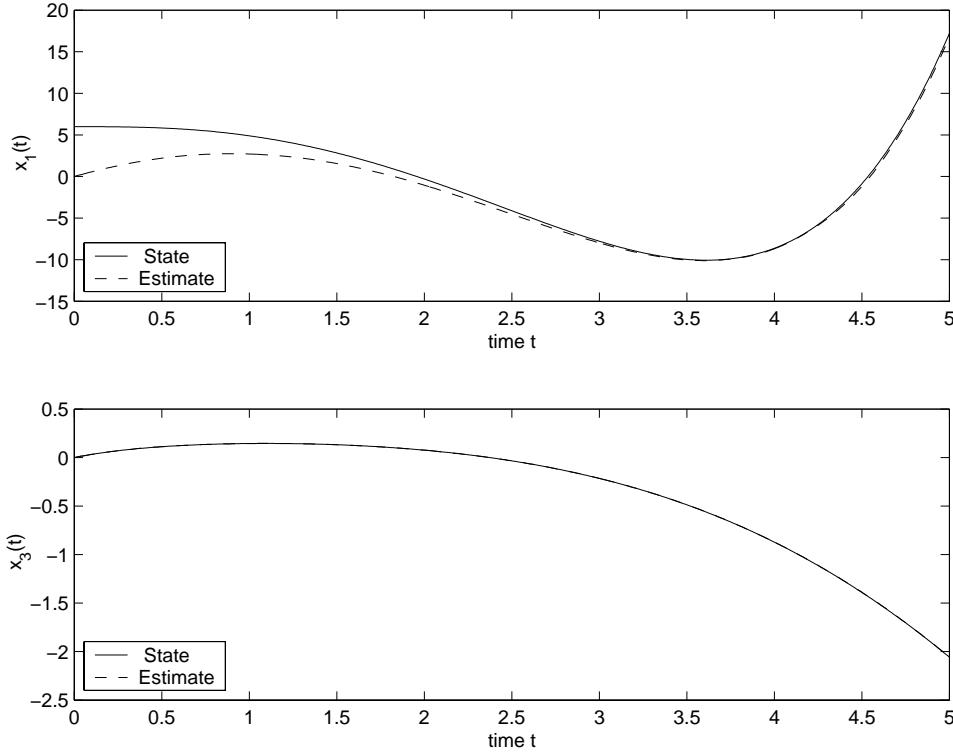


Figure 12.2: The First and Third Variables of the State $x(t)$ and Estimate $\hat{x}(t)$, for Example 12.4.1

12.4.2 Reduced-order State Estimation via Sylvester-observer Equation

As in the case of a full-dimensional observer, a reduced-order observer can also be constructed via solution of a Sylvester-observer equation. The procedure is as follows:

Algorithm 12.4.2 Reduced-order Observer Design via Sylvester-Observer Equation

Inputs: The matrices A , B , and C of order $n \times n$, $n \times m$, and $r \times n$, respectively.

Output: An estimate \hat{x} of the state vector x .

Assumptions: (i) (A, C) is observable. (ii) C is of full rank.

Step 1. Choose an $(n - r) \times (n - r)$ **stable** matrix F .

Step 2. Solve the reduced-order Sylvester-observer equation for a full rank $(n - r) \times n$ solution X :

$$XA - FX = GC,$$

choosing the $(n - r) \times r$ matrix G appropriately. (Numerical methods for solving the Sylvester-observer equation will be described in Section 12.7).

Step 3. Compute $P = XB$

Step 4. Find the $(n - r)$ dimensional reduced-order observer z by solving the system of differential equations:

$$\dot{z} = Fz + Gy + Pu, \quad z(0) = z_0. \quad (12.4.14)$$

Step 5. Find an estimate \hat{x} of x :

$$\hat{x} = \begin{pmatrix} C \\ X \end{pmatrix}^{-1} \begin{pmatrix} y \\ z \end{pmatrix}.$$

Note: If we write $\begin{pmatrix} C \\ X \end{pmatrix}^{-1} = (\bar{S}_1, \bar{S}_2)$, then \hat{x} can be written in the compact form:

$$\hat{x} = \bar{S}_1 y + \bar{S}_2 z. \quad (12.4.15)$$

Example 12.4.2 Consider Example 12.4.1 again.

Step 1. Choose $F = \begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix}$.

Step 2. Choose $G = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$.

The solution X of the Sylvester-observer equation $XA - FX = GC$ is

$$X = \begin{pmatrix} -0.117 & -0.0822 & 62.1322 & 37.2007 \\ -0.1364 & -1.9296 & 428.2711 & -173.4895 \end{pmatrix}.$$

Step 3. $P = XB = \begin{pmatrix} 21.7151 & 1.2672 \\ 149.1811 & 20.4653 \end{pmatrix}$.

Step 4. The 2-dimensional reduced-order observer is given by $\dot{z} = Fz + Gy + Pu$, where F, G , and P are the matrices found in Step 1, Step 2, and Step 3, respectively.

An estimate \hat{x} of x is

$$\hat{x} = \begin{pmatrix} C \\ X \end{pmatrix}^{-1} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} -24.5513 & -135.1240 & 124.1400 & -18.0098 \\ 1 & 0 & 0 & 0 \\ -0.0033 & -0.0360 & 0.0395 & -0.0034 \\ 0 & 0.0175 & 0 & 0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix}$$

(Note that if $\hat{x} = \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{pmatrix}$ and $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, then $\hat{x}_2 = y_1$, $\hat{x}_4 = 0.0175y_2$, same as obtained in Example 12.4.1 using the eigenvalue assignment method).

Comparison of the States and Estimates for Example 12.4.2

Below we compare the actual state vector with the estimated one obtained by Algorithm 12.4.2 on the data of Example 12.4.1. The solid line corresponds to the actual state and the dotted line corresponds to the estimated state. MATLAB function **ode23** was used to solve the underlying differential equations with the same initial conditions as in Example 12.4.1. *The third components are indistinguishable.*

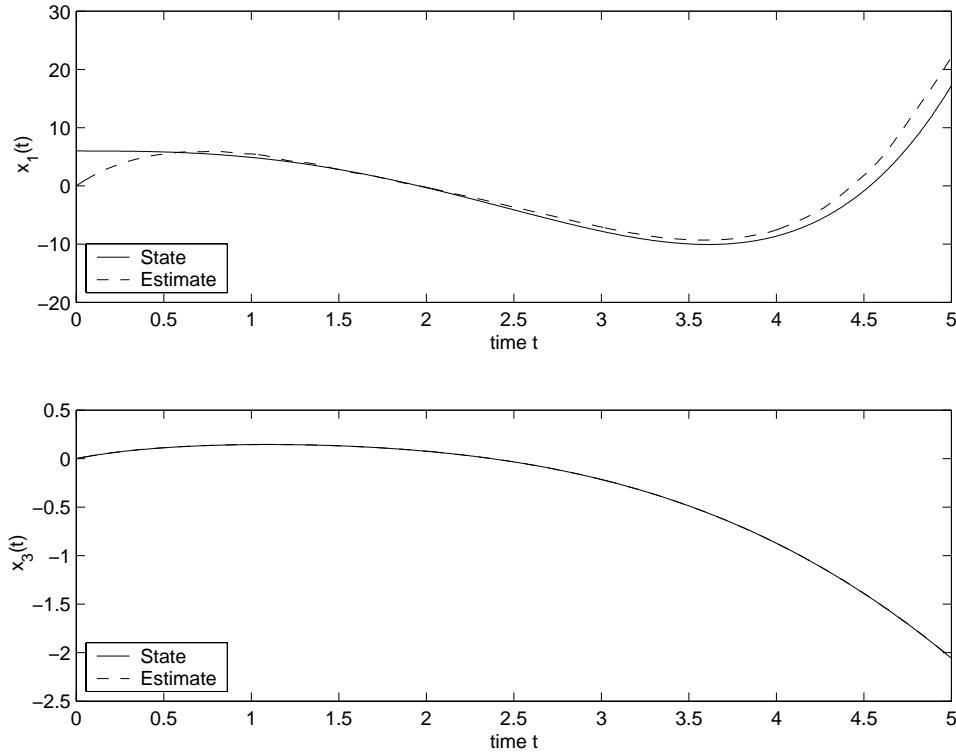


Figure 12.3: The First and Third Variables of the State $x(t)$ and the Estimate $\hat{x}(t)$, for Example 12.4.2.

12.5 Combined State Feedback and Observer Design

When an estimate \hat{x} of x is used in the feedback control law

$$u = s - K\hat{x} \quad (12.5.1)$$

in place of x , one naturally wonders: **what effect will there be on the eigenvalue assignment?** We consider only the reduced-order case, here. The same conclusion, of course, is true for a full-order observer.

Using (12.5.1) in (12.2.1), we obtain

$$\begin{aligned}\dot{x} &= Ax + B(s - K\hat{x}) \\ &= Ax + B(s - K\bar{S}_1y - K\bar{S}_2z) \quad (\text{using (12.4.15)}) \\ &= Ax + B(s - K\bar{S}_1Cx - K\bar{S}_2z) \\ &= (A - BK\bar{S}_1C)x - BK\bar{S}_2z + Bs.\end{aligned}$$

Also, the equation (12.4.14), can be written as

$$\begin{aligned}\dot{z} &= Fz + Gy + Pu = Fz + GCx + P(s - K\bar{S}_1y - K\bar{S}_2z) \\ &= (GC - PK\bar{S}_1C)x + (F - PK\bar{S}_2)z + Ps\end{aligned}$$

(using (12.5.1) and (12.4.15)).

Thus the combined (feedback and observer) system is given by

$$\begin{aligned}\begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} &= \begin{pmatrix} A - BK\bar{S}_1C & -BK\bar{S}_2 \\ GC - PK\bar{S}_1C & F - PK\bar{S}_2 \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + \begin{pmatrix} B \\ P \end{pmatrix} s \\ y &= (C, 0) \begin{pmatrix} x \\ z \end{pmatrix}\end{aligned}\tag{12.5.2}$$

Applying to this system the equivalence transformation, given by the nonsingular matrix $\begin{pmatrix} I & 0 \\ -X & I \end{pmatrix}$, and noting that $e = z - Xx$, $XA - FX = GC$ and $P = XB$, we have, after some algebraic manipulations:

$$\begin{aligned}\begin{pmatrix} \dot{x} \\ \dot{e} \end{pmatrix} &= \begin{pmatrix} A - BK & -BK\bar{S}_2 \\ 0 & F \end{pmatrix} \begin{pmatrix} x \\ e \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} s \\ y &= (C, 0) \begin{pmatrix} x \\ e \end{pmatrix}.\end{aligned}\tag{12.5.3}$$

Thus, the eigenvalues of the combined system are the union of the eigenvalues of the closed-loop matrix $A - BK$ and of the observer matrix F .

Therefore, the observer design and feedback design can be carried out independently; and the calculation of the feedback gain is not affected whether the true state x or the estimated state \hat{x} is used.

This property is known as the **separation property**.

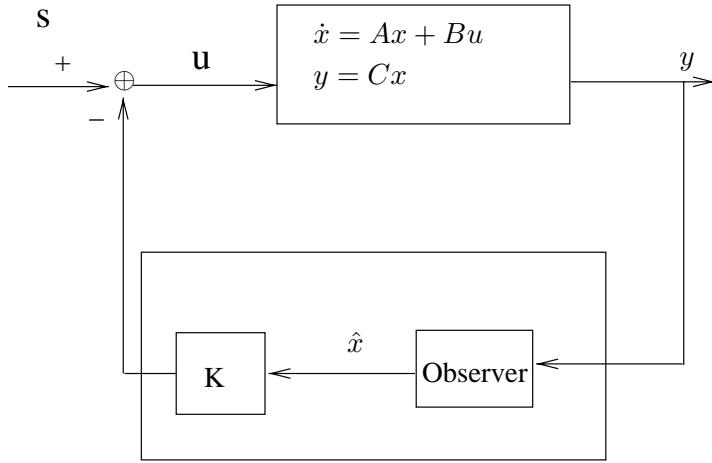


Figure 12.4: Observer-Based State Feedback

12.6 Characterization of Nonsingular Solutions of the Sylvester Equation

We have just seen that the design of an observer via the Sylvester-observer equation requires a nonsingular solution X for the full-order design (**Algorithm 12.3.1**) or a full rank solution X for the reduced-order design (**Algorithm 12.4.2**). In this section, we describe some necessary conditions for a unique solution of the Sylvester equation to have such properties. For the sake of convenience, we consider the full-order case (that is, A and F are $n \times n$) only. The results, however, hold for the reduced-order case also and the proofs given here can be easily modified and are left as an exercise [**Exercise 7**].

The following Theorem was proved by Bhattacharyya and DeSouza (1981). The proof here has been taken from Chen (1984).

Theorem 12.6.1 (Necessary Conditions for Nonsingularity of the Sylvester Equation Solution)

Let A, F, G , and C be, respectively, of order $n \times n, n \times n, n \times r$ and $r \times n$. Let X be a unique solution of the Sylvester equation

$$XA - FX = GC. \quad (12.6.1)$$

Thus necessary conditions for X to be nonsingular are that (A, C) is observable and (F, G) is controllable.

Proof: From the given equation (12.6.1), we have

$$\begin{aligned}
 XA^0 - F^0X &= 0 \quad (\text{Noting that } A^0 = I_{n \times n} \text{ and } F^0 = I_{n \times n}.) \\
 XA - FX &= GC \\
 XA^2 - F^2X &= GCA + FGC \\
 &\vdots \\
 XA^n - F^nX &= GCA^{n-1} + FGCA^{n-2} + \cdots + F^{n-1}GC.
 \end{aligned}$$

Let $a(\lambda) = \lambda^n + a_1\lambda^{n-1} + \cdots + a_n$ be the characteristic polynomial of A , and let's denote the controllability matrix of the pair (F, G) by C_{FG} , and the observability matrix of the pair (A, C) by O_{AC} .

First of all, we note that the uniqueness of X implies that the matrix $a(F)$ is nonsingular and vice versa. This is seen as follows: By Theorem 8.2.1, X is a unique solution of (12.6.1) if and only if A and F do not have a common eigenvalue. Again, A and F do not have a common eigenvalue if and only if the matrix $a(F)$ is nonsingular; because the eigenvalues of $a(F)$ are the n numbers $\prod_{j=1}^n (\mu_i - \lambda_j)$, $i = 1, \dots, n$; where, λ_i 's are the eigenvalues of A and μ_i 's are the eigenvalues of F . Thus, $a(F)$ is nonsingular if and only if X is unique solution of (12.6.1).

Now, multiplying the above equations, respectively, by $a_n, a_{n-1}, \dots, 1$, and using the Cayley-Hamilton Theorem, we obtain after some algebraic manipulations

$$X = -[a(F)]^{-1}C_{FG} R O_{AC}, \quad (12.6.2)$$

where

$$R = \begin{pmatrix} a_{n-1}I & a_{n-2}I & \cdots & a_1I & I \\ a_{n-2}I & a_{n-3}I & \cdots & I & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ a_1I & I & \cdots & 0 & 0 \\ I & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

From (12.6.2), it then immediately follows that for X to be nonsingular, the rectangular matrices C_{FG} and O_{AC} must have full rank; or, in other words, the pair (F, G) must be controllable and the pair (A, C) must be observable. ■

Corollary 12.6.1 *If G is $n \times 1$ and C is $1 \times n$, then necessary and sufficient conditions for the unique solution X of (12.6.1) to be nonsingular are that (F, G) is controllable and (A, C) is observable.*

Proof: In this case, both the matrices C_{FG} and O_{AC} are square matrices. Thus, from (12.6.2), it immediately follows that X is nonsingular if and only if (F, G) is controllable and (A, C) is observable. ■

Theorem 12.6.1 has recently been generalized by Datta, Hong, and Lee (1997) giving a necessary and sufficient condition for nonsingularity of X . We state the result below and refer the readers to the paper for the proof.

Theorem 12.6.2 (Characterization of the Nonsingularity of the Sylvester Equation Solution) Let A, F , and R be $n \times n$ matrices. Let $a(\lambda) = \lambda^n + a_1\lambda^{n-1} + \cdots + a_n$ be the characteristic polynomial of A .

Define

$$\begin{aligned} S &= (F^{n-1} + a_1F^{n-2} + \cdots + a_{n-1}I)R + (F^{n-2} + a_1F^{n-3} + \cdots + a_{n-2}I)RA \\ &\quad + \cdots + (F + a_1I)RA^{n-2} + RA^{n-1} \end{aligned}$$

Then a unique solution X of the Sylvester equation

$$FX - XA = R$$

is nonsingular if and only if S is nonsingular. Furthermore, the unique solution X is given by

$$X = (a(F))^{-1}S.$$

(Note again that the uniqueness of X implies that $a(F)$ is nonsingular). ■

Remarks: The results of Theorems 12.6.1 and 12.6.2 also hold in case the matrix X is not necessarily a square matrix. In fact, this general case has been dealt with in the papers by Bhattacharyya and DeSouza (1981), and Datta, Hong, and Lee (1997), and conditions for the unique solution to have full rank have been derived there.

12.7 Numerical Solutions of the Sylvester-Observer Equation

In this section, we discuss numerical methods for solving the Sylvester-observer equation. These methods are based on the reduction of the observable pair (A, C) to the observer-Hessenberg form (H, \bar{C}) , described in Chapter 6.

The methods use the following template.

Step 1. Reduction of the Problem. The pair (A, C) is transformed to the observer-Hessenberg form by orthogonal similarity; that is, an orthogonal matrix O is constructed such that

$$\begin{aligned} OAO^T &= H, \text{ an unreduced block upper-Hessenberg matrix} \\ CO^T &= \bar{C} = (0, C_1) \end{aligned}$$

The equation $XA - FX = GC$ is then transformed to $XO^T OAO^T - FXO^T = GCO^T$

or

$$YH - FY = G\bar{C}, \tag{12.7.1}$$

where $Y = XO^T$.

Step 2. Solution of the Reduced Problem. The reduced Hessenberg Sylvester-observer equation (12.7.1) is solved.

Step 3. Recovery of the solution X of the original problem. The solution X of the original problem is recovered from the solution of the reduced problem:

$$X = YO \quad (12.7.2)$$

We now discuss the implementation of Step 2. Step 3 is straightforward. Implementation of Step 1 has been described in Chapter 6.

The simplest way to solve the equation (12.7.1) is to choose the matrices F and G completely satisfying the controllability requirement of the pair (F, G) . In that case, the Sylvester-observer equation reduces to an ordinary Sylvester equation, and, therefore, can be solved using the **Hessenberg–Schur method**, described in Chapter 8.

Indeed, F can be chosen in the lower real Schur form (RSF), as required by the method. Therefore, computations will be greatly reduced. We will not repeat the procedure here. Instead, we will present below two simple recursive procedures, designed specifically for solution of the reduced-order Sylvester-observer equation (12.7.1).

12.7.1 A Recursive Method for the Hessenberg Sylvester-Observer Equation.

In the following, we describe a recursive procedure for solving the reduced multi-output Sylvester-observer equation

$$YH - FY = G\bar{C}. \quad (12.7.3)$$

The procedure is due to Van Dooren (1984). The procedure computes simultaneously the matrices F, Y , and G , assuming that (H, \bar{C}) is observable.

Set $q = n - r$ and assume that Y has the form

$$Y = \begin{pmatrix} 1 & y_{12} & \cdots & & \cdots & y_{1,n} \\ \ddots & \ddots & & & & \vdots \\ 0 & & 1 & y_{q,q+1} & \cdots & y_{q,n} \end{pmatrix} \quad (12.7.4)$$

and choose F in lower triangular form (for simplicity):

$$F = \begin{pmatrix} f_{11} & 0 & \cdots & \cdots & 0 \\ f_{21} & f_{22} & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ f_{q1} & \cdots & \cdots & \cdots & f_{qq} \end{pmatrix}, \quad G = \begin{pmatrix} g_1^T \\ g_2^T \\ \vdots \\ g_q^T \end{pmatrix} \quad (12.7.5)$$

where the diagonal entries f_{ii} , $i = 1, \dots, q$ are known and the remaining entries of F are to be found. It has been shown in (Van Dooren (1984)) that a solution Y in the above form always exists. The reduced Sylvester-observer equation can now be solved recursively for Y, F , and G , as follows.

Let g_i^T denote the i th row of G . Comparing the first row of the equation (12.7.3), we obtain

$$(1, y_1)H - f_{11}(1, y_1) = g_1^T \bar{C}. \quad (12.7.6)$$

Similarly, comparing the i th row of that equation, we have

$$(0, 0, \dots, 0, 1, y_i)H - (f_i, f_{ii}, 0, \dots, 0)Y = g_i^T \bar{C}, \quad i = 2, 3, \dots, q \quad (12.7.7)$$

In above, $y_i = (y_{i,i+1}, \dots, y_{i,n})$ and $f_i = (f_{i1}, f_{i2}, \dots, f_{i,i-1})$.

The equations (12.7.6) and (12.7.7) can be, respectively, written as

$$(y_1, g_1^T) \begin{bmatrix} (H - f_{11}I)_{bottom(n-1)} \\ -\bar{C} \end{bmatrix} = -[1st \text{ row of } (H - f_{11}I)] \quad (12.7.8)$$

and

$$(f_i, y_i, g_i^T) \begin{bmatrix} -Y_{top(i-1)} \\ (H - f_{ii}I)_{bottom(n-i)} \\ -\bar{C} \end{bmatrix} = -[ith \text{ row of } (H - f_{ii}I)], \quad (12.7.9)$$

where $Y_{top(i-1)}$ and $(H - f_{ii}I)_{bottom(n-i)}$ denote, respectively, the top $i - 1$ rows of Y and the bottom $n - i$ rows of $H - f_{ii}I$. Because of the structure of the observer–Hessenberg form (H, \bar{C}) , the above systems are consistent and these systems can be solved recursively to compute the unknown entries of the matrices Y , F , and G .

We illustrate how to solve these equations in the special case with $n = 3$, $r = 1$. The reduced equation to be solved in this case is:

$$\begin{pmatrix} 1 & y_{12} & y_{13} \\ 0 & 1 & y_{23} \\ 0 & 0 & h_{32} \end{pmatrix} \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & h_{32} & h_{33} \end{pmatrix} - \begin{pmatrix} f_{11} & 0 \\ f_{21} & f_{22} \end{pmatrix} \begin{pmatrix} 1 & y_{12} & y_{13} \\ 0 & 1 & y_{23} \end{pmatrix} = \underbrace{\begin{pmatrix} g_{11} \\ g_{21} \end{pmatrix}}_G \underbrace{\begin{pmatrix} 0 & 0 & c_1 \end{pmatrix}}_{\bar{C}}.$$

Comparing the first row of the last equation we have

$$\begin{cases} y_{12}h_{21} = f_{11} - h_{11} \\ y_{12}(h_{22} - f_{11}) + y_{13}h_{32} = -h_{12} \\ y_{13}(h_{33} - f_{11}) + y_{12}h_{23} - g_{11}c_1 = -h_{13} \end{cases} \quad (12.7.10)$$

Similarly, comparing the second row, we have

$$\begin{cases} -f_{21} = -h_{21} \\ y_{23}h_{32} - f_{21}y_{12} - f_{22} = -h_{22} \\ y_{23}h_{33} - f_{21}y_{13} - f_{22}y_{23} - g_{21}c_1 = -h_{23} \end{cases} \quad (12.7.11)$$

The system (12.7.10) can be written as

$$(y_{12}, y_{13}, g_{11}) \begin{pmatrix} h_{21} & h_{22} - f_{11} & h_{23} \\ 0 & h_{32} & h_{33} - f_{11} \\ 0 & 0 & -c_1 \end{pmatrix} = \begin{pmatrix} f_{11} - h_{11} \\ -h_{12} \\ -h_{13} \end{pmatrix}^T$$

Similarly, the system (12.7.11) can be written as

$$(f_{21}, y_{23}, g_{21}) \begin{pmatrix} -1 & -y_{12} & -y_{13} \\ 0 & h_{32} & h_{33} - f_{22} \\ 0 & 0 & -c_1 \end{pmatrix} = \begin{pmatrix} -h_{21} \\ f_{22} - h_{22} \\ -h_{23} \end{pmatrix}^T.$$

Note that since the pair (A, C) is observable, h_{21} , h_{32} , and c_1 are different from zero and, therefore, the matrices of the above two systems are nonsingular.

Algorithm 12.7.1 A Recursive Algorithm for the Multi-Output Sylvester-Observer Equation

Inputs: The matrices $A_{n \times n}$, and $C_{r \times n}$.

Output: A full-rank solution X of the reduced-order Sylvester-observer equation

$$XA - FX = GC.$$

Assumption: (A, C) is observable.

Step 0. Set $n - r = q$.

Step 1. Transform the pair (A, C) to the observer-Hessenberg pair (H, \bar{C}) :

$$\begin{aligned} OAO^T &= H \\ CO^T &= \bar{C} \end{aligned}$$

Step 2. Construct $F = (f_{ij})$ as a $q \times q$ lower triangular matrix, where the diagonal entries f_{ii} , $i = 1, \dots, q$ are arbitrarily give numbers, and the off-diagonal entries are to be computed.

Step 3. Solve for Y satisfying

$$YH - FY = G\bar{C},$$

where Y has the form (12.7.4), as follows:

Compute the first row of Y and the first row of G by solving the system (12.7.8). Compute the second through q th rows of Y , the second through q th rows of F , and the second through q th rows of G simultaneously, by solving the system (12.7.9).

Step 4. Recover X from Y :

$$X = YO.$$

Example 12.7.1 Consider Example 12.4.1 again.

Here $n = 4, r = 2$.

Step 1. The observer-Hessenberg pair of (A, C) is given by:

$$H = \begin{pmatrix} -0.0200 & 2.4000 & 0.0050 & -32.0000 \\ 0 & -1.6000 & 0.0180 & 1.2000 \\ -0.1400 & -1.3000 & 0.4400 & -30.0000 \\ 0 & 1.000 & 0 & 0 \end{pmatrix},$$

$$\bar{C} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 57.3 \end{pmatrix}.$$

The transforming matrix

$$O = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Step 2. Let's choose $f_{11} = -1, f_{22} = -2$

Step 3. The solution of the system (12.7.8) is $(0, 7, 6.7, 10.085, -4.1065)$.

Thus $y_1 = (0, 7, 6.7), g_1 = (10.085, -4.1065)$. The first row of $Y = (1, 0, 7, 6.7)$.

The solution of the system (12.7.9) is $(0.0007, -0.0053, -0.4068, 0, 0.0094)$.

Thus $f_{21} = 0.0007$ and

$$y_2 = (-0.0053, -0.4068),$$

$$g_2 = (0, 0.0094).$$

$$\text{So, } F = \begin{pmatrix} -1 & 0 \\ 0.0007 & -2 \end{pmatrix}, G = \begin{pmatrix} 10.085 & -4.1065 \\ 0 & 0.0094 \end{pmatrix}.$$

The second row of $Y = (0, 1, -0.0053, -0.4068)$.

Therefore,

$$Y = \begin{pmatrix} 1 & 0 & 7 & 6.7 \\ 0 & 1 & -0.0053 & -0.4068 \end{pmatrix}.$$

Step 4. Recover X from $Y : X = YO = \begin{pmatrix} 1 & 7 & 0 & 6.7 \\ 0 & -0.0053 & 1 & -0.4068 \end{pmatrix}$.

Verify: $\|XA - FX - GC\|_2 = O(10^{-14})$

Flop-count:

Solving for F, G , and Y (using the special structures of these matrices): $2(n - r)rn^2$ flops.

Obtaining the observer-Hessenberg form: $2(3n + r)n^2$ flops (including the construction of O).

Recovering X from $Y : 2(n - r)n$

Total: (**About**) $(6 + 2r)n^3$.

MATCONTROL Note: Algorithm 12.7.1 has been implemented in MATCONTROL function **sylvobsm**.

12.7.2 A Recursive Block-Triangular Algorithm for the Hessenberg Sylvester-Observer Equation

A block version of Algorithm 12.7.1 has recently been obtained by Carvalho and Datta (2001). This block algorithm seems to be computationally slightly more efficient than Algorithm 12.7.1 and is suitable for high-performance computing. We describe this new block algorithm below. As in Algorithm 12.7.2, assume that the observable pair (A, C) has been transformed to an observer-Hessenberg pair (H, \bar{C}) , that is, an orthogonal matrix O has been computed such that

$$OAO^T = H \text{ and } \bar{C} = C O^T = \begin{bmatrix} 0 & \dots & 0, & C_1 \end{bmatrix}$$

where $H = (H_{ij})$ is block upper Hessenberg with diagonal blocks $H_{ii} \in \mathbb{R}^{n_i \times n_i}$, $i = 1, 2, \dots, p$ and $n_1 + \dots + n_p = n$.

Given the Observer-Hessenberg pair (H, \bar{C}) , we now show how to compute the matrices Y, F and G in **blocks** such that

$$YH - FY = G\bar{C}. \quad (12.7.12)$$

Partitioning the matrices F, Y and G conformably with H allows us to write the above equation as

$$\begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1p} \\ & Y_{22} & \dots & Y_{2p} \\ & & Y_{qq} & Y_{qp} \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} & \dots & H_{1p} \\ H_{21} & H_{22} & \dots & H_{2p} \\ & H_{32} & \dots & H_{3p} \\ & & H_{p-1,p} & H_{pp} \end{bmatrix} - \begin{bmatrix} F_{11} \\ F_{21} & F_{22} \\ F_{q1} & \dots & F_{qq} \end{bmatrix} \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1p} \\ & Y_{22} & \dots & Y_{2p} \\ & & Y_{qq} & Y_{qp} \end{bmatrix} = \begin{bmatrix} G_1 \\ \dots \\ G_q \end{bmatrix} \begin{bmatrix} 0 & 0 & \dots & 0 & C_1 \end{bmatrix}.$$

We set $Y_{ii} = I_{r \times r}, i = 1, 2, \dots, q$ for simplicity. Since matrix F is required to have a preassigned spectrum \mathcal{S} , we distribute the elements of \mathcal{S} among the diagonal blocks of F in such a way that $\Omega(F) = \mathcal{S}$, where $\Omega(M)$ denotes the spectrum of M . A complex conjugate pair is distributed as a 2×2 matrix and a real one as a 1×1 scalar on the diagonal of F . Note that, some compatibility between the structure of \mathcal{S} and the parameters $n_i, i = 1, \dots, p$ is required to exist for this to be possible.

Equating now the corresponding blocks on left and right hand sides we obtain:

$$\sum_{k=i}^{j+1} Y_{ik} H_{kj} - \sum_{k=1}^{\min(i,j)} F_{ik} Y_{kj} = 0, j = 1, 2, \dots, p-1 \quad (12.7.13)$$

$$\sum_{k=i}^p Y_{ik} H_{kp} - \sum_{k=1}^i F_{ik} Y_{kp} = G_i C_1. \quad (12.7.14)$$

From (12.7.13) and (12.7.14) we conclude

$$F_{ij} = 0 \text{ for } j = 1, 2, \dots, i-2, \text{ and } F_{ij} = H_{ij} \text{ for } j = i-1.$$

Thus, equations (12.7.13) and (12.7.14) are reduced to

$$\sum_{k=i}^{j+1} Y_{ik} H_{kj} - \sum_{k=\max(i-1,1)}^i F_{ik} Y_{kj} = 0, \quad j = i, i+1, \dots, p-1 \quad (12.7.15)$$

$$\sum_{k=i}^p Y_{ik} H_{kp} - \sum_{k=\max(i-1,1)}^i F_{ik} Y_{kp} = G_i C_1, \quad \text{for } i = 1, 2, \dots, q. \quad (12.7.16)$$

For a computational purpose we rewrite equation (12.7.15) as

$$\sum_{k=i}^j Y_{ik} H_{kj} + Y_{i,j+1} H_{j+1,j} - \sum_{k=\max(i-1,1)}^i F_{ik} Y_{kj} = 0, \quad j = i, i+1, \dots, p-1$$

that is, for $j = i, i+1, \dots, p-1$,

$$Y_{i,j+1} H_{j+1,j} = - \sum_{k=i}^j Y_{ik} H_{kj} + \sum_{k=\max(i-1,1)}^i F_{ik} Y_{kj} \quad (12.7.17)$$

Equations (12.7.16) and (12.7.17) allow us to compute the off-diagonal blocks Y_{ij} of Y and the blocks G_i of G recursively.

This is illustrated in the following, in the special case when $p = 4, q = 3$:

First row : $i = 1$

$$H_{11} + Y_{12} H_{21} - F_{11} = 0 \text{ (Solve for } Y_{12})$$

$$H_{12} + Y_{12} H_{22} + Y_{13} H_{32} - F_{11} Y_{12} = 0 \text{ (Solve for } Y_{13})$$

$$H_{13} + Y_{12} H_{23} + Y_{13} H_{33} + Y_{14} H_{43} - F_{11} Y_{13} = 0 \text{ (Solve for } Y_{14})$$

$$H_{14} + Y_{12} H_{24} + Y_{13} H_{34} + Y_{14} H_{44} - F_{11} Y_{14} = G_1 C_1 \text{ (Solve for } G_1)$$

Second row : $i = 2$

$$H_{22} + Y_{23} H_{32} - F_{21} Y_{12} - F_{22} = 0 \text{ (Solve for } Y_{23})$$

$$H_{23} + Y_{23} H_{33} + Y_{24} H_{43} - F_{21} Y_{13} - F_{22} Y_{23} = 0 \text{ (Solve for } Y_{24})$$

$$H_{24} + Y_{23} H_{34} + Y_{24} H_{44} - F_{21} Y_{14} - F_{22} Y_{24} = G_2 C_1 \text{ (Solve for } G_2)$$

Third row : $i = 3$

$$H_{33} + Y_{34} H_{43} - F_{32} Y_{23} - F_{33} = 0 \text{ (Solve for } Y_{34})$$

$$H_{34} + Y_{34} H_{44} - F_{32} Y_{24} - F_{33} Y_{34} = G_3 C_1 \text{ (Solve for } G_3)$$

The above discussion leads to the following algorithm:

Algorithm 12.7.2: A Recursive Block Triangular Algorithm for the Multi-output Sylvester Observer Equation.

Input: Matrices $A \in \mathbb{R}^{n \times n}$, $C \in \mathbb{R}^{r \times n}$ of full-rank and the self-conjugate set $\mathcal{S} \in \mathbb{C}^{n-r}$.

Output: Block matrices X, F and G , such that $\Omega(F) = \mathcal{S}$ and $XA - FX = GC$.

Step 1: Reduce (A, C) to observer-Hessenberg form (H, \bar{C}) . Let $n_i, i = 1, \dots, p$ be the dimension of the diagonal blocks H_{ii} of the matrix H .

Step 2: Partition matrices Y, F and G in blocks according to the block structure of H . Let $q = p - 1$.

Step 3: Distribute the elements of \mathcal{S} along the diagonal blocks $F_{ii}, i = 1, 2, \dots, q$ such that $\Omega(F) = \mathcal{S}$; the complex conjugate pairs as 2×2 blocks and the real ones as 1×1 scalars along the diagonal of the matrix F .

Step 4: Set $Y_{11} = I_{n_1 \times n_1}$.

Step 5: For $i = 2, 3, \dots, q$ set

$$F_{i,i-1} = H_{i,i-1}, \quad Y_{ii} = I_{n_i \times n_i}$$

Step 6: For $i = 1, 2, \dots, q$ do

Step 6.1: For $j = i, i+1, \dots, p-1$, solve the upper triangular system for $Y_{i,j+1}$:

$$Y_{i,j+1} H_{j+1,j} = - \sum_{k=i}^j Y_{ik} H_{kj} + \sum_{k=\max(i-1,1)}^i F_{ik} Y_{kj}$$

Step 6.2: Solve the triangular system for G_i :

$$G_i C_1 = \sum_{k=i}^p Y_{ik} H_{kp} - \sum_{k=\max(i-1,1)}^i F_{ik} Y_{kp}$$

Step 7: Form the matrices Y, F and G from their the computed blocks.

Step 8: Recover $X = Y O$.

Return

Remark: Recall that once the matrix X is obtained, the estimated state-vector $\hat{x}(t)$ can be computed from

$$\begin{bmatrix} C \\ X \end{bmatrix} \hat{x}(t) = \begin{bmatrix} y(t) \\ z(t) \end{bmatrix}$$

It is interesting to note that the matrix X does not need to be computed explicitly for this purpose, because, the above system is equivalent to:

$$\begin{bmatrix} \bar{C} \\ Y \end{bmatrix} \hat{x}(t) = \begin{bmatrix} y(t) \\ z(t) \end{bmatrix} O^T.$$

The matrix $\begin{pmatrix} \bar{C} \\ Y \end{pmatrix}$ is a nonsingular block upper Hessenberg by the construction of Y . This structure is very important from the computational point of view since it can possibly be exploited in high performance computations.

Flop-count and Comparison of Efficiency:

Flop-count of Algorithm 12.7.2.

1. Reduction to observer Hessenberg form using the staircase algorithm:

$$6n^3 + 2rn^2 \text{ flops}$$

2. Computation of Y using Steps 4 - 8 of the algorithm:

$$\begin{aligned} \sum_{i=1}^{p-1} \sum_{j=i}^p [(j-i+1)(2r^3) + 2(2r^3) + r^2] &= \sum_{i=1}^{p-1} \sum_{j=i}^p \{[2(j-i)+7]r^3 + r^2\} \\ &\approx \sum_{i=1}^{p-1} [(p-i)^2 + (p-i)] r^3 \approx \left[\frac{(p-1)p(2p-1)}{6} + \frac{(p-1)p}{2} \right] r^3 \\ &\approx \frac{n^3}{3} + \frac{rn^2}{2} \text{ flops.} \end{aligned}$$

3. Computation of X from Y : n^3 flops (note that the matrix Y is a unit block triangular matrix).

Thus total count is $\frac{19n^3}{3} + \frac{5r}{2}n^2$ flops.

Comparison of Efficiency.

Algorithm 12.7.1 requires about $(6 + 2r)n^3$ flops. [Note: the flop count given in Van Dooren (1984) is nearly one half of that given here; this is because a "flop" is counted there as a multiplication / division coupled with an addition / subtraction.]

Also, it can be shown that a recent block algorithm of Datta and Sarkissian (2000) requires about $52n^3/3$ flops.

*Thus Algorithm 12.7.2 is more than three times faster than both Van Dooren's (**Algorithm 12.7.1**) and the Datta-Sarkissian algorithms.*

Besides, this algorithm is suitable for implementations using the recently developed and widely-used scientific computing software package LAPACK (Anderson et al.(1999)), since it is composed of BLAS-3 operations such as matrix-matrix multiplications, QR factorizations and solutions of triangular systems with multiple right hand sides.

Example 12.7.2: We consider Example 12.7.1 again,

Step 1: The matrices H , \bar{C} , and O are given by:

$$H = \begin{bmatrix} -0.0200 & 2.4000 & 0.0050 & -32.0000 \\ 0 & -1.6000 & 0.0180 & 1.2000 \\ -0.1400 & -1.3000 & 0.4400 & -3.0000 \\ 0 & 1.0000 & 0 & 0 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 1.0000 & 0 \\ 0 & 57.3000 \end{bmatrix},$$

$$O = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \bar{C} = (0, C_1).$$

Step 2: $q = 1$.

Steps 3 and 4:

$$F = F_{11} = \begin{bmatrix} -1.00 & 0 \\ 0 & -2.00 \end{bmatrix}, Y_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Step 5: skipped ($q = 1$).

Step 6: $i = 1$:

Step 6.1: $j = 1$. Solve the triangular system $Y_{12}H_{21} = -Y_{11}H_{11} + F_{11}Y_{11}$ for Y_{12} :

$$Y_{12} = \begin{bmatrix} 7.0000 & 6.7000 \\ 0 & -0.4000 \end{bmatrix}$$

Step 6.2: Solve triangular system $G_1C_1 = Y_{11}H_{12} + Y_{12}H_{22} - F_{11}Y_{12}$ for G_1 :

$$G_1 = \begin{bmatrix} 10.0850 & -0.8080 \\ 0.0180 & 0.0070 \end{bmatrix}$$

Step 7 : Form matrices Y, F and G from the computed blocks:

$$Y = \begin{bmatrix} 1 & 0 & 6.0000 & 6.7000 \\ 0 & 1 & 0 & -0.4000 \end{bmatrix}$$

$$F = \begin{bmatrix} -1.000 & 0 \\ 0 & -2.000 \end{bmatrix}, G = \begin{bmatrix} 10.0850 & -0.8080 \\ 0.0180 & 0.0070 \end{bmatrix}$$

Step 8: Recover $X = YO$:

$$X = \begin{bmatrix} 1 & 6.0000 & 0 & 6.7000 \\ 0 & 0 & 1 & -0.4000 \end{bmatrix}$$

Verify: $\|XA - FX - GC\|_2 = 5.5511 \times 10^{-17}$ and $\Omega(T) = \{-2.0000, -1.0000\}$. Thus the residue is small and the spectrum of F has been assigned accurately.

MATCONTROL Note: Algorithm 12.7.3 has been implemented in MATCONTROL function **sylvobsmb**.

Comparison of the State and Estimate for Example 12.7.2

Figure 12.5 shows the relative error between the exact state $x(t)$ and the estimate $\hat{x}(t)$ satisfying

$$\begin{bmatrix} \bar{C} \\ Y \end{bmatrix} \hat{x}(t) = \begin{bmatrix} y(t) \\ z(t) \end{bmatrix} O^T$$

with the data above and $u(t)$ as the unit step function. The underlying systems of ordinary differential equations were solved by using **MATLAB** procedure **ode45** with zero initial conditions. The relative error is defined by $\frac{\|x(t) - \hat{x}(t)\|_2}{\|x(t)\|_2}$.

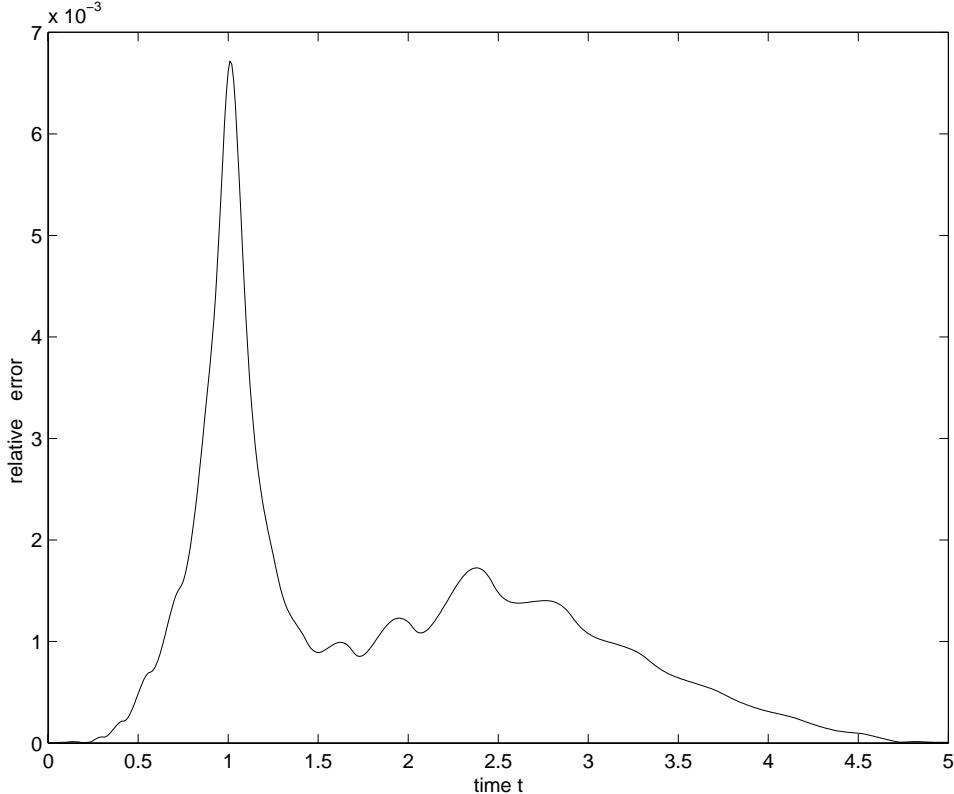


Figure 12.5: Relative Error Between the State and Estimate

12.8 Numerical Solution of a Constrained Sylvester-observer Equation

In this section, we consider the problem of solving a constrained reduced-order Sylvester-observer equation. Specifically, the following problem is considered:

Solve the reduced-order Sylvester-observer equation

$$XA - FX = GC \quad (12.8.1)$$

such that

$$XB = 0 \quad (12.8.2)$$

and

$$\begin{bmatrix} X \\ C \end{bmatrix} \quad (12.8.3)$$

has full rank.

The importance of solving the constrained Sylvester equation lies in the fact that if the constraint (12.8.2) is satisfied then the feedback system with the reduced-order observer has the same robustness properties as that of the direct feedback system (see Tsui (1988)).

We state a recent method of Barlow, Monahemi, and O'Leary (1992) to solve the above problem. A basic idea behind the method is to transform the given equation to a reduced-order unconstrained equation and then recover the solution of the constrained equation from that of the reduced unconstrained equation. We skip the details and present below just the algorithm. For details of the development of the algorithm, see the above paper by Barlow et al. (1992).

Algorithm 12.8.1 An Algorithm for Constrained Sylvester-observer Equation

Inputs: (1) The system matrices A, B , and C of order $n \times n, n \times m$, and $r \times n$, respectively.
(2) A matrix F of order $(n - r)$.

Output: An $(n - r) \times n$ matrix X and an $(n - r) \times r$ matrix G satisfying (12.8.1) such that

$\begin{pmatrix} X \\ C \end{pmatrix}$ is nonsingular and $XB = 0$.

Assumptions: (A, C) is observable, $n > r > m$, and $\text{rank}(CB) = \text{rank}(B) = m$.

Step 1. Find the QR factorization of B :

$$B = W \begin{pmatrix} S \\ 0 \end{pmatrix},$$

where S is $m \times m$, upper triangular and has full rank, and W is $n \times n$ and orthogonal.

Partition $W = (W_1, W_2)$, where W_1 is $n \times m$ and W_2 is $n \times (n - m)$.

Step 2. Set

$$\begin{aligned} A_1 &= W_2^T A W_1 \\ A_2 &= W_2^T A W_2 \\ C_1 &= C W_1 \\ C_2 &= C W_2 \end{aligned}$$

Step 3. Find a QR factorization of C_1 :

$$C_1 = (Q_1, Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where Q_1 is $r \times m$, Q_2 is $r \times (r - m)$, and R is an $m \times m$ upper triangular matrix with full rank.

Step 4. Define E by

$$E = \begin{pmatrix} E_1 \\ E_2 \end{pmatrix} = Q^T C_2,$$

where E_1 is $m \times (n - m)$, E_2 is $(r - m) \times (n - m)$ and $Q = (Q_1, Q_2)$.

Step 5. Form $\hat{A} = A_2 - A_1 R^{-1} E_1$. Solve the Sylvester equation:

$$Z\hat{A} - FZ = G_2 E_2,$$

choosing G_2 randomly.

Step 6. Set $G_1 = Z A_1 R^{-1} = Z J$, $G = (G_1, G_2) Q^T$, and $X = Z W_2^T$.

(Note that Z is of order $(n - r) \times (n - m)$ and $J = A_1 R^{-1}$ is computed by solving the upper triangular system $J R = A_1$).

MATHCONTROL Note. Algorithm 12.8.1 has been implemented in MATCONTROL functions **sylvobsc**.

Example 12.8.1 Consider solving the Equation (12.8.1) using Algorithm 12.8.1 with

$$A = \begin{pmatrix} -0.02 & 0.005 & 2.4 & -3.2 \\ -0.14 & 0.44 & -1.3 & -3 \\ 0 & 0.018 & -1.6 & 1.2 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 57.3 \end{pmatrix}, \text{ and}$$

$$F = \begin{pmatrix} -1 & 0 \\ 1 & -2 \end{pmatrix}.$$

Then $n = 4$, $r = 2$, $m = 1$.

Step 1. $W_1 = (-0.5, -0.5, -0.5, -0.5)^T$.

$$W_2 = \begin{pmatrix} -0.5 & -0.5 & -0.5 \\ 0.8333 & -0.1667 & -0.1667 \\ -0.1667 & 0.8333 & -0.1667 \\ -0.1667 & -0.1667 & 0.8333 \end{pmatrix}.$$

$$S = -2.$$

$$\text{Step 2. } A_1 = W_2^T A W_1 = \begin{pmatrix} 1.5144 \\ -0.2946 \\ -0.9856 \end{pmatrix},$$

$$A_2 = W_2^T A W_2 = \begin{pmatrix} 0.9015 & -1.6430 & 0.5596 \\ 0.1701 & -2.5976 & 2.9907 \\ -0.4185 & -0.2230 & 1.5604 \end{pmatrix},$$

$$C_1 = C W_1 = \begin{pmatrix} -0.5 \\ -28.65 \end{pmatrix},$$

$$C_2 = CW_2 = \begin{pmatrix} 0.8333 & -0.1667 & -0.1667 \\ -9.55 & -9.55 & 47.75 \end{pmatrix}.$$

Step 3. $Q = \begin{pmatrix} -0.0174 & -0.9998 \\ -0.9998 & 0.0174 \end{pmatrix}.$

$$R = 28.6544$$

Step 4. $E = Q^T C_2 = \begin{pmatrix} 9.5340 & 9.5515 & -47.7398 \\ -0.9998 & 0 & 0.9998 \end{pmatrix}.$

$$E_2 = \begin{pmatrix} -0.9998 & 0 & 0.9998 \end{pmatrix}$$

Step 5. $\hat{A} = A_2 - A_1 R^{-1} E_1 = \begin{pmatrix} 0.3976 & -2.1478 & 1.96 \\ -0.0721 & -2.4944 & 2.4999 \\ -0.0905 & 0.1056 & -0.0817 \end{pmatrix}.$

Choose

$$G_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The solution Z of the Sylvester equation: $Z\hat{A} - FZ = G_2 E_2$

$$Z = \begin{pmatrix} -0.6715 & 0.9589 & -0.0860 \\ -0.2603 & -0.5957 & 0.9979 \end{pmatrix}$$

Step 6. $G_1 = Z A_1 R^{-1} = \begin{pmatrix} -0.0424 \\ -0.0420 \end{pmatrix}.$

$$G = (G_1, G_2) Q^T = \begin{pmatrix} -0.9991 & 0.0598 \\ 0.0007 & 0.0419 \end{pmatrix},$$

$$X = \begin{pmatrix} -0.1007 & -0.7050 & 0.9254 & -0.1196 \\ -0.0709 & -0.2839 & -0.6199 & 0.9743 \end{pmatrix}.$$

Verify:

(i) $\| XA - FX - GC \| = O(10^{-15}).$

(ii) $XB = 10^{-16} \begin{pmatrix} 0.4163 \\ 0.4163 \end{pmatrix}.$

(iii) $\text{rank} \begin{pmatrix} X \\ C \end{pmatrix} = 4.$

Note: If G_2 were chosen as

$$G_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Then the solution X would be rank-deficient and consequently $\begin{pmatrix} X \\ C \end{pmatrix}$ would be also rank-deficient. Indeed, in this case, $X = \begin{pmatrix} -0.1006 & -0.7044 & 0.9246 & -0.1195 \\ -0.1006 & -0.7044 & 0.9246 & -0.1195 \end{pmatrix}$, which has rank 1.

12.9 Optimal State Estimation: The Kalman Filter

So far we have discussed the design of an observer ignoring the “noise” in the system; that is, we assumed that all the inputs were given exactly and all the outputs were measured exactly without any errors. But in a practical situation, the measurements are always corrupted with noise. Therefore, it is more practical to consider a system with noise. In this section, we consider the problem of finding the optimal steady state estimation of the states of a **stochastic system**. Specifically, the following problem is addressed.

Consider the stochastic system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + Fw(t) \\ y(t) &= Cx(t) + v(t), \end{aligned} \tag{12.9.1}$$

where $w(t)$ and $v(t)$ represent “noise” in the input and the output, respectively. The problem is to find the linear estimate $\hat{x}(t)$ of $x(t)$ from all past and current output $\{y(s), s \leq t\}$ that minimizes the mean square error:

$$E[\|x(t) - \hat{x}(t)\|^2], \text{ as } t \rightarrow \infty. \tag{12.9.2}$$

The following assumptions are made:

1. The system is **controllable** and **observable**. (12.9.3)

Note that the controllability assumption implies that the noise $w(t)$ excites all modes of the system and the observability implies that the noiseless output $y(t) = Cx(t)$ contains information about all states.

2. Both w and v are white noise, **zero-mean** stochastic processes.

That is, for all t and s ,

$$Ew(t) = 0, \quad Ev(t) = 0, \tag{12.9.4}$$

$$Ew(t)w^T(s) = W\delta(t-s), \tag{12.9.5}$$

$$Ev(t)v^T(s) = V\delta(t-s), \tag{12.9.6}$$

where W and V are **symmetric and positive definite covariance matrices**, and $\delta(t-s)$ is the Dirac delta function.

3. The noise processes w and v are **uncorrelated** with one another; that is,

$$Ew(t)v^T(s) = 0. \quad (12.9.7)$$

4. The initial state x_0 is a **Gaussian zero-mean** random variable with known covariance matrix; and uncorrelated with w and v . That is,

$$\begin{aligned} E[x_0] &= 0 \\ E[x_0x_0^T] &= S, \quad E[x_0w^T(t)] = 0, \quad E[x_0v^T(t)] = 0, \end{aligned} \quad (12.9.8)$$

where S is the positive semidefinite covariance matrix.

The following is a well-known (almost classical) result on the solution of the above problem using an algebraic Riccati equation. For a proof, see Kalman and Bucy (1961). For more details on this topic, see Kailath, Sayed, and Hassibi (2000).

Theorem 12.9.1 *Under the assumptions (12.9.3)-(12.9.8), the best estimate $\hat{x}(t)$ (in the linear least-mean-square sense) can be generated by the **Kalman filter**:*

$$\dot{\hat{x}}(t) = (A - K_f C)\hat{x}(t) + Bu(t) + K_f y(t), \quad \hat{x}(0) = \hat{x}_0 \quad (12.9.9)$$

where $K_f = X_f C^T V^{-1}$, and X_f is the symmetric positive definite solution of the algebraic Riccati equation:

$$AX + XA^T - XC^T V^{-1} CX + FWF^T = 0. \quad (12.9.10)$$

■

Definition 12.9.1 *The matrix $K_f = X_f C^T V^{-1}$ is called the **filter gain matrix**.*

Note: The output estimate $\hat{y}(t)$ is given by $\hat{y}(t) = C\hat{x}(t)$.

The error between the measured output $y(t)$ and the predicted output $C\hat{x}(t)$ is given by the residual $r(t)$:

$$r(t) = y(t) - C\hat{x}(t).$$

where \hat{x} is generated by (12.9.9).

Algorithm 12.9.1 *The State Estimation of the Stochastic System Using Kalman Filter*

- Inputs:**
1. The matrices A, B, C , and F defining the system (12.9.1)
 2. The covariance matrices V and W (both symmetric and positive definite).

Output: An estimate $\hat{x}(t)$ of $x(t)$ such that $E[\|x(t) - \hat{x}(t)\|^2]$ is minimized, as $t \rightarrow \infty$.

Assumptions: (12.9.3)-(12.9.8).

Step 1. Obtain the unique symmetric positive definite solution X_f of the algebraic Riccati equation:

$$AX_f + X_f A^T - X_f C^T V^{-1} C X_f + F W F^T = 0.$$

Step 2. Form the filter gain matrix $K_f = X_f C^T V^{-1}$.

Step 3. Obtain the estimate $\hat{x}(t)$ by solving (12.9.9).

Duality Between Kalman Filter and the LQR Problems

The algebraic Riccati equation (12.9.10) in Theorem 12.9.1 is dual to the CARE that arises in the solution of the LQR problem. To distinguish it from the CARE, it will be referred to as the **Continuous-time Filter Algebraic Riccati Equation** (CFARE). Using this duality, the following important properties of the Kalman filter, *dual to those of the LQR problem described in Chapter 10*, can be established [Exercise 15].

1. Guaranteed Stability

The filter matrix $A - K_f C$ is stable; that is, $\operatorname{Re}\lambda_i(A - K_f C) < 0$; $i = 1, 2, \dots, n$, where $\lambda_i, i = 1, \dots, n$, are the eigenvalues of $A - K_f C$.

2. Guaranteed Robustness

Let V be a diagonal matrix and let $W = I$. Let $G_{KF}(s)$ and $G_{FOL}(s)$ denote, respectively, the Kalman-filter loop-transfer matrix and the filter open-loop transfer matrix (from $w(t)$ to $y(t)$); that is,

$$G_{KF}(s) \equiv C(sI - A)^{-1} K_f, \quad (12.9.11)$$

and

$$G_{FOL}(s) \equiv C(sI - A)^{-1} F. \quad (12.9.12)$$

Then the following equality holds:

$$(I + G_{KF}(s))V(I + G_{KF}(s))^* = V + G_{FOL}(s)G_{FOL}^*(s). \quad (12.9.13)$$

Using the above equality, one obtains

$$(I + G_{KF}(s))(I + G_{KF}(s))^* \geq I. \quad (12.9.14)$$

In terms of singular values, one then can deduce that

$$\sigma_{\min}(I + G_{KF}(s)) \geq 1 \quad (12.9.15)$$

or

$$\sigma_{\max}(I + G_{KF}(s))^{-1} \leq 1$$

and

$$\sigma_{\min}(I + G_{KF}^{-1}(s)) \geq \frac{1}{2}. \quad (12.9.16)$$

See the article by Athans on Kalman filtering in the Control Handbook (1996, 589-594), edited by W. S. Levine, IEEE Press/CRC Press.

Example 12.9.1 Consider the stochastic system:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) + w(t) \\ y(t) &= Cx(t) + v(t)\end{aligned}$$

with A, B , and C as in Example 12.4.1.

Take $W = BB^T$, $V = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $F = I_{4 \times 4}$

Step 1. The symmetric positive definite solution X_f of the CFARE

$$AX + XA^T - XC^T V^{-1} CX + FWF^T = 0$$

is

$$X_f = \begin{pmatrix} 8.3615 & 0.0158 & 0.0187 & -0.0042 \\ 0.0158 & 9.0660 & 0.0091 & -0.0031 \\ 0.0187 & 0.0091 & 0.0250 & 0.0040 \\ -0.0042 & -0.0031 & 0.0040 & 0.0016 \end{pmatrix}$$

Step 2. The filter gain matrix $K_f = X_f C^T V^{-1}$ is

$$K_f = \begin{pmatrix} 0.0158 & -0.2405 \\ 9.0660 & -0.1761 \\ 0.0091 & 0.2289 \\ -0.0031 & 0.0893 \end{pmatrix}$$

The optimal state estimator of $\hat{x}(t)$ is given by

$$\dot{\hat{x}}(t) = (A - K_f C) \hat{x}(t) + Bu(t) + K_f y(t).$$

The filter eigenvalues, that is the eigenvalues of $A - K_f C$, are $\{-0.0196, -8.6168, -3.3643 \pm j2.9742\}$.

MATLAB Note. The MATLAB function **kalman** designs a Kalman state estimator given the state-space model and the process and noise covariance data. **kalman** is available in MATLAB **Control System Toolbox**.

Comparison of the State and the Estimate for Example 12.9.1

Below we compare the state with the estimate one obtained in Example 12.9.1 with $x(0) = \hat{x}(0) = (-6 \ -1 \ 1 \ 2)^T$ and $u(t) = H(t)(1 \ 1 \ 1 \ 1)^T$, $H(t)$ is the unit step function. Only the first and second variables are compared. The solid line corresponds to the exact state and the dotted line corresponds to the estimated state. *The graphs of the second variables are indistinguishable.*

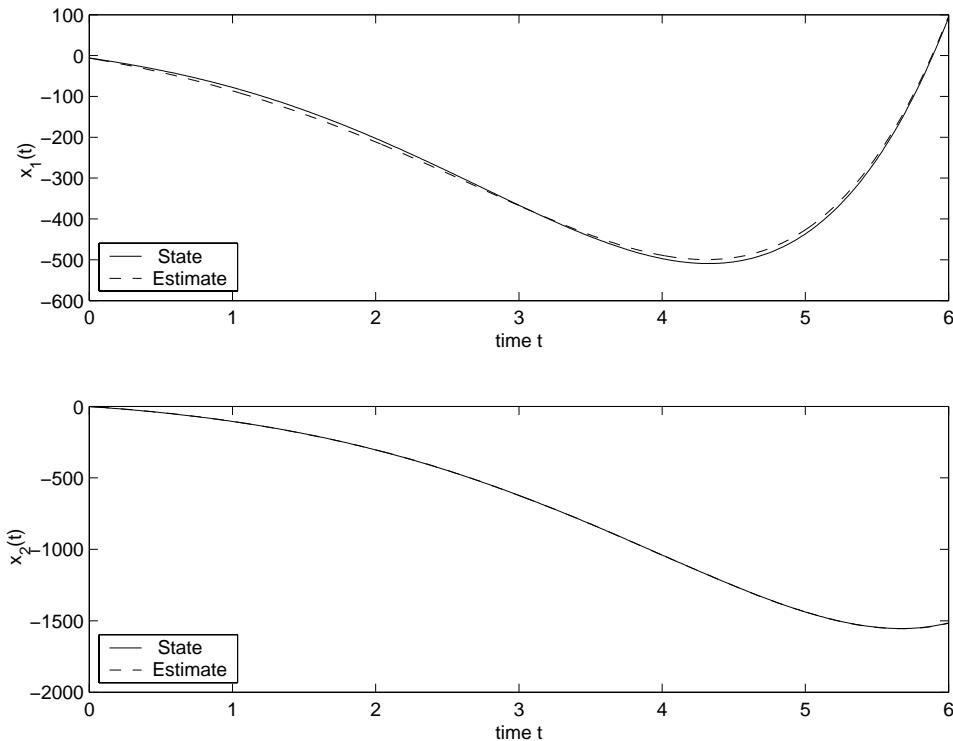


Figure 12.6: The First and Second Variables of the State $x(t)$ and Estimate $\hat{x}(t)$, obtained by Kalman Filter.

The Kalman Filter for the Discrete-time System

Consider now the discrete stochastic system:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Fw_k \\ y_k &= Cx_k + v_k, \end{aligned} \quad (12.9.17)$$

where w , and v are the process and measurement noise. Then, under the same assumptions as made in the continuous-time case, it can be shown that the state error covariance is minimized in steady-state when the filter gain is given by

$$K_d = X_d C^T (C X_d C^T + V)^{-1}, \quad (12.9.18)$$

where X_d is the symmetric positive definite solution of the discrete-Riccati equation:

$$X = A(X - XC^T(CXC^T + V)^{-1}CX)A^T + FWF^T, \quad (12.9.19)$$

and V and W are the symmetric positive definite covariance matrices, that is,

$$E(v_k v_j^T) = V\delta_{kj}, \quad E(w_k w_j^T) = W\delta_{kj}; \quad (12.9.20)$$

and

$$\delta_{kj} = \begin{cases} 0 & \text{if } k \neq j \\ 1 & \text{if } k = j \end{cases} \quad (12.9.21)$$

Definition 12.9.2 In analogy with the continuous-time case, the discrete algebraic Riccati equation (12.9.19), arising in discrete Kalman filter will be called the discrete filter algebraic Riccati equation or DFARE, for short.

12.10 The Linear Quadratic Gaussian Problem

The LQR problems deal with optimization of a performance measure for a deterministic system. The **Linear Quadratic Gaussian** (LQG) problems deal with optimization of a performance measure for a stochastic system.

Specifically, the **continuous-time Linear Quadratic Gaussian** (LQG) problem is defined as follows:

Consider the controllable and observable stochastic system (12.9.1) and the quadratic objective function

$$J_{QG} = \lim_{T \rightarrow \infty} \frac{1}{2T} E \left[\int_{-T}^T (x^T Q x + u^T R u) dt \right],$$

where the weighting matrices Q and R are, respectively, symmetric positive semidefinite and positive definite. Suppose that the noise $w(t)$ and $v(t)$ are both Gaussian, white, zero-mean, and stationary processes with positive definite covariance matrices W and V . The problem is to find the optimal control $u(t)$ that minimizes the average cost.

Solution of the LQG Problem via Kalman Filter

The solution of the LQG problem is obtained by combining the solutions of the deterministic LQR problem and the optimal state estimation problem using the Kalman filter (see the next subsection on the separation property of the LQG design).

The control vector $u(t)$ for the LQG problem is given by

$$u(t) = -K_c \hat{x}(t) \quad (12.10.1)$$

where

- (i) the matrix K_c is the feedback matrix of the associated LQR problem; that is,

$$K_c = R^{-1}B^T X_c, \quad (12.10.2)$$

X_c satisfying the CARE: $X_c A + A^T X_c + Q - X_c B R^{-1} B^T X_c = 0$

- (ii) the vector $\hat{x}(t)$ is generated by the Kalman filter:

$$\dot{\hat{x}}(t) = (A - K_f C) \hat{x}(t) + B u(t) + K_f y(t). \quad (12.10.3)$$

The filter gain matrix $K_f = X_f C^T V^{-1}$ and X_f satisfies the CFARE

$$AX_f + X_f A^T - X_f C^T V^{-1} C X_f + F W F^T = 0. \quad (12.10.4)$$

The minimum value of the performance measure J_{QG} is given by

$$J_{QG}^* = \text{trace}(X_c K_f V K_f^T) + \text{trace}(X_f Q), \quad (12.10.5)$$

where X_c is the stabilizing solution of the CARE, X_f is the stabilizing solution of the CFARE and $K_f = X_f C^T V^{-1}$.

For a proof of the above, see Dorato et al. (1995).

The LQG design via Kalman filter is illustrated in Figure 12.7.

■

The LQG Separation Property: In this section we establish the LQG separation property. For the sake of convenience, we assume that $F = I$. By substituting (12.10.1) into (12.10.3), we obtain the compensator

$$\begin{aligned} \dot{\hat{x}}(t) &= (A - BK_c - K_f C) \hat{x}(t) + K_f y(t) \\ u(t) &= -K_c \hat{x}(t) \end{aligned} \quad (12.10.6)$$

The transfer function $M(s)$ of this compensator (from $y(t)$ to $u(t)$) can be easily written down:

$$M(s) = -K_c(sI - A + BK_c + K_f C)^{-1} K_f \quad (12.10.7)$$

From (12.10.6) and (12.9.1), it is easy to see that the closed-loop matrix satisfies the differential equation

$$\begin{pmatrix} \dot{x}(t) \\ \dot{\hat{x}}(t) \end{pmatrix} = \begin{pmatrix} A & -BK_c \\ K_f C & A - BK_c - K_f C \end{pmatrix} \begin{pmatrix} x(t) \\ \hat{x}(t) \end{pmatrix} + \begin{pmatrix} I & O \\ O & K_f \end{pmatrix} \begin{pmatrix} w(t) \\ v(t) \end{pmatrix} \quad (12.10.8)$$

Define the error vector

$$e(t) = x(t) - \hat{x}(t). \quad (12.10.9)$$

Then from (12.10.8) and (12.10.9), we obtain

$$\begin{pmatrix} \dot{x}(t) \\ \dot{e}(t) \end{pmatrix} = \begin{pmatrix} A - BK_c & BK_c \\ O & A - K_f C \end{pmatrix} \begin{pmatrix} x(t) \\ e(t) \end{pmatrix} + \begin{pmatrix} I & O \\ I & -K_f \end{pmatrix} \begin{pmatrix} w(t) \\ v(t) \end{pmatrix}.$$

Thus, the $2n$ closed-loop eigenvalues are the union of the n eigenvalues of $A - BK_c$ and the n eigenvalues of $A - K_f C$.

Furthermore, if (A, B) is controllable and (A, C) is observable, then both the matrices $A - BK_c$ and $A - K_f C$ are stable.

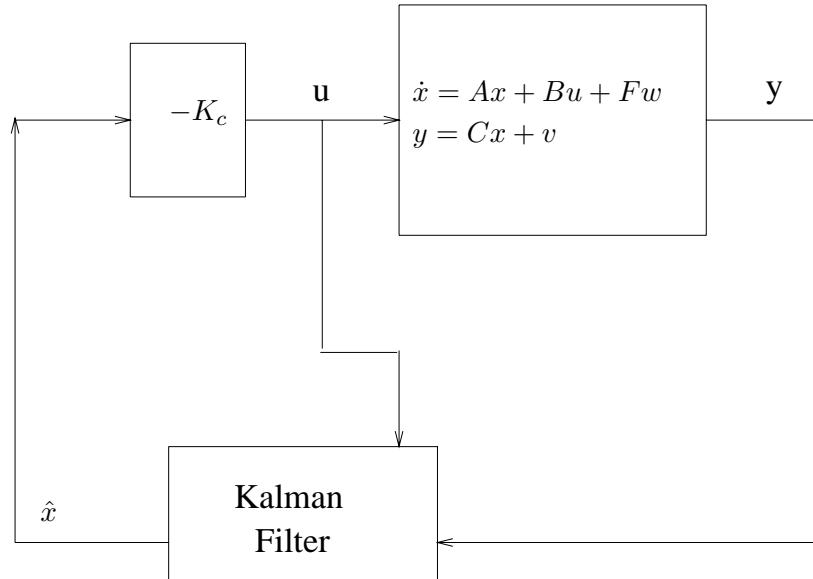


Figure 12.7: The LQG Design via Kalman Filter

Algorithm 12.10.1 The Continuous-time LQG Design Method

Inputs:

1. The matrices A, B, C , and F defining the system (12.9.1).
2. The covariance matrices V and W .

Output: The control vector $u(t)$ generated by the LQG regulator.

Assumptions: (12.9.3)-(12.9.8).

Step 1. Obtain the symmetric positive definite stabilizing solution X_c of the CARE:

$$XA + A^T X - XBR^{-1}B^T X + Q = 0. \quad (12.10.10)$$

Step 2. Compute $K_c = R^{-1}B^T X_c$

Step 3.

3.1. Solve the CFARE:

$$AX + XA^T - XC^T V^{-1} CX + FWF^T = 0 \quad (12.10.11)$$

to obtain the symmetric positive definite stabilizing solution X_f .

3.2. Compute filter gain matrix

$$K_f = X_f C^T V^{-1} \quad (12.10.12)$$

Step 4. Solve for $\hat{x}(t)$:

$$\dot{\hat{x}}(t) = (A - BK_c - K_f C)\hat{x}(t) + K_f y(t), \quad \hat{x}(0) = \hat{x}_0. \quad (12.10.13)$$

Step 5. Determine the control law

$$u(t) = -K_c \hat{x}(t) \quad (12.10.14)$$

Remark: Though the optimal closed-loop system will be asymptotically stable, the LQG design method described above does not have the same properties as the LQR design method; in fact, most of the nice properties of the LQR design are lost by the introduction of the Kalman filter. See Doyle (1978) and Zhou et al. (1996, pp. 398-399).

Overall, the LQG design has lower stability margins than the LQR design and its sensitivity properties are not as good as those of the LQR design.

It might be possible to recover some of the desirable properties of the LQR design by choosing the weights appropriately. This is known as the **Loop Transfer Recovery** (LTR). The details are out of the scope of this book. See Doyle and Stein (1979, 1981) and the book by Anderson and Moore (1990).

Example 12.10.1 We consider the LQG design for the helicopter problem of Example 12.9.1, with

$$Q = C^T C, \quad \text{and} \quad R = I_{2 \times 2},$$

and the same W and V .

Step 1. The stabilizing solution X_c of the CARE (computed by MATLAB function **are**) is

$$X_c = \begin{pmatrix} 0.0071 & -0.0021 & -0.0102 & -0.0788 \\ -0.0021 & 0.1223 & 0.0099 & -0.1941 \\ -0.0102 & 0.0099 & 41.8284 & 174.2 \\ -0.0788 & -0.1941 & 174.2 & 1120.9 \end{pmatrix}$$

Step 2. The control gain matrix K_c is

$$K_c = R^{-1}B^T X_c = \begin{pmatrix} -0.0033 & 0.0472 & 14.6421 & 60.8894 \\ 0.0171 & -1.0515 & 0.2927 & 3.2469 \end{pmatrix}$$

Step 3. The filter gain matrix K_f computed in Example 12.9.1 is $K_f = \begin{pmatrix} 0.0158 & -0.2405 \\ 9.0660 & -0.1761 \\ 0.0091 & 0.2289 \\ -0.0031 & 0.0893 \end{pmatrix}$.

The Closed-loop Eigenvalues: The closed-loop eigenvalues are the union of the eigenvalues of $A - BK_c$ (the controller eigenvalues) and those of $A - K_f C$ (the filter eigenvalues):

$$\{-3.3643 \pm 2.9742j, -0.0196, -8.6168\} \cup \{-0.0196, -8.6168, -3.3643 \pm 2.9742j\}.$$

The minimum Value of J_{QG} : $J_{QG}^* = 42.5327$.

MATLAB Note: The MATLAB function (from the **control system toolbox**) **lqgreg** forms the LQG regulator by combining the Kalman estimator designed with **Kalman** and the optimal state-feedback gain designed with **lqr**. In case of a discrete-time system, the command **dlqr** is used in place of **lqr**.

Cascade Realization

An alternative technique to solve the LQG problem is to compute the feedback transfer matrix $M(s)$ that links the output y to the input u , defined in (12.10.1).

This is known as the **LQG Cascade realization**. Details can be found in Dorato et al. (1995). Note that $A - BK_c$ and $A - K_f C$ are separately stable, but the matrix $A - BK_c - K_f C$ is not necessarily stable. Thus the cascade realization may require an unstable compensator, whereas the estimator realization will always lead to a stable compensator. For details, see Dorato et al. (1995).

Example 12.10.2 We consider the helicopter example again with the same data of the estimator realization. $M(s)$ is given by

$$M(s) = -C_f(sI - A_f)^{-1}B_f,$$

where $A_f = A - BK_c - K_f C$, $C_f = K_c$, and $B_f = K_f$. With the K_f and K_c as computed in Example 12.10.1, we have

$$A_f = \begin{pmatrix} -0.0175 & -0.1436 & 0.3852 & -26.3518 \\ 0.0084 & -17.6863 & -4.0536 & -13.9065 \\ 0.0010 & 0.0018 & -6.7274 & -33.2584 \\ 0 & 0.0031 & 1 & -5.1191 \end{pmatrix},$$

The minimum value of J_{QG} is $28.3476 + 14.1851 = 42.5327$.

12.11 Some Selected Software

12.11.1 MATLAB CONTROL SYSTEM TOOLBOX

LQG design tools

- `kalman` - Kalman estimator
- `kalmd` - Discrete Kalman estimator for continuous plant
- `lqgreg` - Form LQG regulator given LQ gain and Kalman estimator

12.11.2 MATCONTROL

- `SYLVOBSC` - Solving the constrained multi-output Sylvester-observer equation
- `SYLVOBSM` - Solving the multi-output Sylvester-observer equation
- `SYLVOBSMB` - Block triangular algorithm for the multi-output Sylvester-observer equation

12.11.3 CSP-ANM

Design of reduced-order state estimator (observer)

- The reduced-order state estimator using pole assignment approach is computed by `ReducedOrderEstimator [system, poles]`.
- The reduced-order state estimator via solution of the Sylvester-observer equation using recursive bidiagonal scheme is computed by `ReducedOrderEstimator [system, poles, Method → RecursiveBidiagonal]` and `ReducedOrderEstimator [system, poles, Method → RecursiveBlockBidiagonal]` (block version of the recursive bidiagonal scheme).
- The reduced-order state estimator via solution of the Sylvester-observer equation using recursive triangular scheme is computed by `ReducedOrderEstimator [system, poles, Method → RecursiveTriangular]` and `ReducedOrderEstimator [system, poles, Method → RecursiveBlockTriangular]` (block version of the recursive triangular scheme).

12.11.4 SLICOT

- `FB01RD` Time-invariant square root covariance filter (Hessenberg form)
- `FB01TD` Time-invariant square root information filter (Hessenberg form)
- `FB01VD` One recursion of the conventional Kalman filter
- `FD01AD` Fast recursive least-squares filter.

12.11.5 MATRIX_X

Purpose: Calculate optimal state estimator gain matrix for a discrete time system.

Syntax: `[EVAL, KE]=DESTIMATOR (A, C, QXX, QYY, QXY) OR`
`[EVAL, KE, P]=DESTIMATOR (A, C, QXX, QYY, QXY)`

Purpose: Calculate optimal state estimator gain matrix for a continuous time system.

Syntax: [EVAL, KE]=ESTIMATOR (A, C, QXX, QYY, QXY)
[EVAL, KE, P]=ESTIMATOR (A, C, QXX, QYY, QXY)

Purpose: Given a plant and optimal regulator, this function designs an estimator which recovers loop transfer robustness via the design parameter RHO. Plots of singular value loop transfer response are made for the (regulator) and (estimator+regulator) systems.

Syntax:

[SC, NSC, EVE, KE, SLTF, NSLTF]=LQELTR (S, NS, QXX, QYY, KR, RHO, WMIN, WMAX,
{ NPTS } , { OPTION }); OR
[SC, NSC, EVE, KR, SLTF, NSLTF]=LQRRLTR (S, NS, RXX, RUU, KE, RHO, OMEGA,
{ OPTION });

Purpose: Given a plant and optimal estimator, this function designs a regulator which recovers loop transfer robustness via the design parameter RHO. Plots of singular value loop transfer response are made for the (estimator) and (regulator+estimator) systems.

Syntax:

[SC, NSC, EVR, KR, SLTF, NSLTF]=LQRRLTR (S, NS, RXX, RUU, KE, RHO, WMIN, WMAX,
{ NPTS } , { OPTION }); OR
[SC, NSC, EVR, KR, SLTF, NSLTF]=LQRRLTR (S, NS, RXX, RUU, KE, RHO, OMEGA,
{ OPTION });

12.12 Summary and Review

In Chapters 10 and 11 we have discussed feedback stabilization, eigenvalue assignment and related problems. Solutions of these problems require that the states are available for measurements. Unfortunately, in many practical situations, all the states are not accessible. One therefore needs to estimate the states by knowing only input and output. This gives rise to **state estimation** problem, which is the subject matter of this Chapter.

I. Full State Estimation

The states can be estimated using

- Eigenvalue assignment approach (**Theorem 12.2.1**)

- Solving the associated Sylvester-like matrix equation, called the **Sylvester-observer equation (Algorithm 12.3.1)**.

In “**the eigenvalue assignment approach**,” the states x can be estimated by constructing the observer

$$\dot{\hat{x}}(t) = (A - KC)\hat{x}(t) + Ky(t) + Bu(t),$$

where the matrix K is constructed such that $A - KC$ is a stable matrix, so that the error $e(t) = x(t) - \hat{x}(t) \rightarrow 0$ as $t \rightarrow \infty$.

Using “**the Sylvester equation approach**”, the states are estimated by solving the Sylvester-observer equation

$$XA - FX = GC,$$

where the matrix F is chosen to be a stable matrix and G is chosen such that the solution X is nonsingular. The estimate $\hat{x}(t)$ is given by

$$\hat{x}(t) = X^{-1}z(t),$$

where $z(t)$ satisfies $\dot{z}(t) = Fz(t) + Gy(t) + XBu(t)$.

II. Reduced-order state Estimation. If the matrix C has full rank r , then the full state estimation problem can be reduced to the problem of estimating only the $n - r$ states.

Again, two approaches: **the eigenvalue assignment approach** and **the Sylvester-observer matrix equation** can be used for reduced-order state estimation.

Reduced-order state estimation via eigenvalue assignment (**Algorithm 12.4.1**) is discussed in **Section 12.4.1**. Here the eigenvalue assignment problem to be solved is of order $n - r$.

In the Sylvester equation approach for reduced-order state estimation, one solves a reduced-order equation

$$XA - FX = GC$$

by choosing F as an $(n - r) \times (n - r)$ stable matrix and choosing G as an $(n - r) \times r$ matrix such that the solution matrix X has full rank. The procedure is described in **Algorithm 12.4.2**.

Two numerical methods for the multi-output equation, both based on reduction of the pair (A, C) to the observer-Hessenberg pair (H, \bar{C}) , are proposed to solve the above reduced-order Sylvester-observer equation. These methods are described in Section 12.7 (**Algorithms 12.7.1 and 12.7.2**).

III. Optimal State Estimation: The Kalman Filter

If there is “noise” in the system, then one has to consider the state-estimation problem for a **stochastic system**. The optimal steady-state estimation of a stochastic system is traditionally done by constructing the **Kalman filter**.

For the continuous-time stochastic system (12.9.1), the Kalman filter is given by

$$\dot{\hat{x}}(t) = (A - K_f C)\hat{x}(t) + Bu(t) + K_f y(t),$$

where $K_f = X_f C^T V^{-1}$ and X_f is the symmetric positive definite solution of the continuous-time filter algebraic Riccati equation (CFARE): $AX + XA^T - XC^T V^{-1} CX + FWF^T = 0$.

The matrices V and W are the covariance matrices associated with “noise” in the output and input, respectively. The matrix K_f is called the **Kalman filter gain**.

It can be shown that under the assumptions (12.9.3)-(12.9.8), the above Riccati equation has a symmetric positive definite solution and the estimate $\hat{x}(t)$ is such that

$$E[\|x(t) - \hat{x}(t)\|^2]$$

is minimized as $t \rightarrow \infty$.

Like the LQR design, the Kalman filter also possess the **guaranteed stability** and **robustness properties**:

- The matrix $A - K_f C$ is stable.
- $\sigma_{\min}(I + G_{KF}(s)) \geq 1$
- $\sigma_{\min}(I + G_{KF}^{-1}(s)) \geq \frac{1}{2}$,

where $G_{KF}(s) = C(sI - A)^{-1}K$.

For the discrete-time system, the discrete-time filter algebraic Riccati equation (DFARE) to be solved is

$$X = A(X - XC^T(CXC^T + V)^{-1}CX)A^T + FWF^T$$

and the discrete **Kalman filter** gain is given by

$$K_d = X_d C^T(CX_d C^T + V)^{-1},$$

where X_d is the stabilizing solution of the above discrete Riccati equation (DFARE).

IV. The Linear Quadratic Gaussian (LQG) Problem

The LQG problem is the problem of finding an optimal control that minimizes a performance measure given a **stochastic** system. **Thus, it is the counterpart of the deterministic LQR problem for a stochastic system.**

Given the stochastic system (12.9.1) and the performance measure J_{QG} , the optimal control $u(t)$ for the LQG problem can be computed as

$$u(t) = -K_c \hat{x}(t),$$

where $K_c = R^{-1}B^T X_c$, X_c being the solution of the CARE arising in the solution of the deterministic LQR problem. The estimate $\hat{x}(t)$ is determined by using the Kalman filter. Specifically, $\hat{x}(t)$ satisfies

$$\dot{\hat{x}}(t) = (A - K_f C)\hat{x}(t) + Bu(t) + K_f y(t),$$

where K_f is the Kalman filter gain computed using the stabilizing solution of the CFARE.

Thus, the LQG problem is solved by first solving the LQR problems followed by constructing a Kalman filter.

Unfortunately, the LQG design described as above does not have some of the nice properties of the LQR problem that we have seen before in Chapter 10. They are lost by the introduction of the Kalman filter.

12.13 Chapter Notes and Further Reading

State estimation is one of the central topics in control systems design and has been discussed in many books (Chen (1984), Kailath (1980), Anderson and Moore (1990), etc.). The idea of reduced-order observers was first originated by Luenberger (1964, 1966, 1971, 1979). The treatment of Section 12.4 on the reduced-order estimation has been taken from Chen (1984).

The term “**Sylvester-observer equation**” was first introduced by the author (Datta (1994)). Algorithm 12.7.1 is due to Van Dooren (1984). Algorithm 12.7.2 is due to Carvalho and Datta (2001). For large-scale solution of this equation, see Datta and Saad (1991); for computing an orthogonal solution to the Sylvester-observer equation, see Datta and Hetti (1997). For a discussion of the numerical properties of the method in Datta and Saad (1991), see Calvetti, Lewis and Reichel (2001). A parallel algorithm for the multi-output Sylvester-observer equation appears in Bischof, Datta and Purkayastha (1996). For numerical solution of the Sylvester-observer equation with F as the Jordan canonical form, see Tsui (1993) and the references therein. For other algorithms for this problem see Datta (1989) and Datta and Sarkissian (2000). The last paper contains an algorithm for designing a “**functional observer**”, which can be used to compute the feedback control law $y = K\hat{x}(t)$ without any matrix inversion.

The method for the constrained Sylvester-observer equation presented in Section 12.8 is due to Barlow, Monahemi and O’Leary (1992). For numerical methods dealing with nearly singular constrained Sylvester-observer equation, see Ghavimi and Laub (1996).

The topic of Kalman filter is by now a classical topic. Since the appearance of the pioneering papers by Kalman (1960), Kalman and Bucy (1961), and Kalman (1964), many books and papers have been written on the subject (see, e.g., Anderson and Moore (1979), Maybeck (1979)), Lewis (1986, 1992), Kwakernaak and Sivan (1972), etc.).

A special issue of IEEE Transactions on Automatic Control, edited by Athans (1971b) was published on the topic of LQG design, which contains many important earlier papers in this area and an extensive bibliography on this subject until 1971. See Dorato et al. (1995) for

up-to-date references. For applications of LQG design see McLean (1990). Gangsaas (1986). Bernstein and Haddad (1989) have discussed LQG control with H_∞ performance bound.

We have not discussed in detail the stability and robustness properties of the LQG design. See the papers of Safonov and Athans (1977) and Doyle (1978) in this context.

For discussions on the LQG loop transfer recovery, see the original paper of Doyle and Stein (1979) and the survey of Stein and Athans (1987), and Section 7.2 of the recent book by Dorato et al. (1995).

EXERCISES

1. Consider Example 5.2.5 with the following data: $M = 2, m = 1, g = 0.18$, and $l = 1$. Take $C = (1, 1, 1)$.
 - (i) Find a feedback matrix K such that the closed-loop matrix $A - BK$ has the eigenvalues $-1, -2, -3, -4$.
 - (ii) Assuming now that the state x is not available for feedback, construct a full-dimensional observer using (a) the eigenvalue assignment method and (b) the Sylvester-observer equation. Compare the results by plotting the error between the true and observed states.
 - (iii) Construct a three-dimensional reduced-order observer using (a) the eigenvalue assignment method and (b) the Sylvester-observer equation. Compare the results by plotting the error between the true and observed states.

In each case (ii) and (iii), choose the observer eigenvalues to be three times as those of the matrix $A - BK$.

2. Are the conditions of Theorem 12.3.1 also necessary? Give reasons for your answer.
3. Prove that the pair (A, C) is observable if and only if the pair $(\bar{A}_{22}, \bar{A}_{12})$ is observable, where \bar{A}_{12} and \bar{A}_{22} are given by (12.4.2).
4. Establish the “**separation property**” stated in Section 12.5 for a full-dimensional observer.
5. Prove that the transfer function matrix of the combined system (12.5.2) of the state feedback and observer can be computed from

$$\dot{x} = (A - BK)x + Br, \quad y = Cx$$

and the transfer function matrix is

$$\hat{G}(s) = C(sI - A + BK)^{-1}B.$$

How do you interpret this result?

6. Construct an example to show that the necessary conditions for the unique solution X of the Sylvester equation stated in Theorem 12.6.1 are not sufficient.
7. Using the ideas from the proof of Theorem 12.6.1 prove that necessary conditions for the existence of a unique full rank solution X in $XA - FX = GC$ such that $T = \begin{pmatrix} C \\ X \end{pmatrix}$ is nonsingular are that (A, C) is observable and (F, G) is controllable.

Prove further that for the single-output case ($r = 1$), the conditions are sufficient as well.

8. Deduce Theorem 12.6.1 from Theorem 12.6.2.
9. Establish the relation (12.7.11).
10. Prove that the eigenvalue assignment approach and the Sylvester-observer equation approach, both for full-dimensional and reduced-order state-estimation, are mathematically equivalent.
11. Compare flop-count of Algorithm 12.4.1 with that of Algorithm 12.4.2. (To implement Step 3 of Algorithm 12.4.1, assume that Algorithm 11.3.1 has been used, and to implement Step 2 of Algorithm 12.4.2, assume that Algorithm 12.7.2 has been used).
12. **(Functional Estimator)** (Chen (1984), pp. 369)). Consider the problem of finding an estimator of the form

$$\begin{aligned}\dot{z}(t) &= Fz(t) + Gy(t) + Hu(t) \\ w(t) &= Mz(t) + Ny(t),\end{aligned}$$

where M and N are row vectors, so that $w(t)$ will approach $kx(t)$ for a constant row vector k , as $t \rightarrow \infty$.

- (a) Show that if M and N are chosen so as to satisfy the equation:

$$MT + N\bar{C} = \bar{k},$$

with T given by

$$\begin{aligned}T\bar{A} - FT &= G\bar{C} \\ H &= T\bar{B},\end{aligned}$$

where \bar{A} and \bar{B} are the same as in (12.4.2), and $\bar{C} = CS^{-1}$, $\bar{k} = kS^{-1}$, and F is a stable matrix, then $w(t)$ will approach $kx(t)$ as $t \rightarrow \infty$.

- (b) Based on the result in (a), formulate an algorithm for designing such an estimator and apply your algorithm to Example 12.4.1.
13. Prove that if (A, C) is observable, then a state-estimator for the discrete-time system

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k\end{aligned}$$

may be constructed as

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L(y_k - C\hat{x}_k),$$

where L is such that the eigenvalues of $A - LC$ have moduli less than 1.

14. Show that for a “deadbeat” observer, that is for an observer with the “observer eigenvalues” equal to zero, the observer state equals the original state.

15. Establish the “**Guaranteed Stability**” and “**Guaranteed robustness**” properties of the Kalman Filter, stated in Section 12.9.
16. Construct an example of cascade realization that needs an unstable compensator, though the open-loop system is stable.
17. Design an experiment to show that the LQG design has lower stability margins than the LQR design.
18. Rework Algorithm 12.4.1 using the QR decomposition of the matrix C , so that the explicit inversion of the matrix S can be avoided.

References

1. K.T. Alfriend, Special Section on Robust Control Design for a Benchmark Problem, *AIAA J. Guidance, Control and Dynamics*, vol. 15, pp.1060-1149, 1992.
2. B.D.O. Anderson and J.B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice Hall, Englewood Cliffs, NJ, 1990.
3. B.D.O. Anderson and J.B. Moore, *Optimal Filtering*, Prentice Hall, Englewood Cliffs, NJ, 1979.
4. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd Edition, SIAM, Philadelphia, 1999.
5. M. Athans, The role and use of the stochastic linear quadratic Gaussian problem in control system design, *IEEE Trans. Automat. Control*, vol. AC-16, pp.529-552, 1971a.
6. M. Athans (Ed.), Special issue on linear-quadratic-Gaussian problem, *IEEE Trans. Automat. Control*, vol. AC-16, 1971b.
7. J. B. Barlow, M. M. Monahemi and D. P. O'Leary, Constrained matrix Sylvester equations, *SIAM J. Matrix Anal. Appl.*, vol. 13, pp.1-9, 1992.
8. D. S. Bernstein and W. M. Haddad, LQG Control with \mathcal{H}^∞ performances bound: A Riccati equation approach, *IEEE Trans. Auto. Control*, vol. AC-34, pp.293-305, 1989.
9. S. P. Bhattacharyya and E. DeSouza, Controllability, observability and the solution of $AX - XB = C$, *Lin. Alg. Appl.* vol. 39, pp.167-188, 1981.
10. J. P. Birdwell and A. J. Laub, Balanced singular values for LQG/LTR design, *Int. J. Control* vol. 45, pp.939-950, 1986.
11. C. Bischof, B. N. Datta and A. Purkayastha, A parallel algorithm for the Sylvester-observer equation, *SIAM J. Sci. Comput.*, vol. 17, no. 3, pp.686-698, 1996.
12. D. Calvetti, B. Lewis and L. Reichel, On the solution of large Sylvester-observer equation, *Num. Lin. Alg. Appl.*, pp. 435-451, 2001.
13. J. Carvalho and B.N. Datta, A block algorithm for the Sylvester-Observer equation arising in state-estimation, *Proc. IEEE Conf. Dec. Control*, Orlando, Florida, 2001.
14. C.-T. Chen, *Linear System Theory and Design*, CBS College Publishing, New York, 1984.
15. *The Control Handbook*, edited by William S. Levine, CRC Press and IEEE Press, Boca Raton, Florida, 1995.

16. R. Y. Chiang and M. G. Safonov, *Robust-Control Toolbox for Use with MATLAB*, Math Works, Natick, MA, 1988.
17. B. N. Datta, Parallel and large-scale matrix computations in control; some ideas, *Lin. Alg. Appl.*, vol. 12, pp.243-264, 1989.
18. B. N. Datta, Linear and numerical linear algebra in control theory: Some research problems, *Lin. Alg. Appl.*, vol. 197/198, pp.755-790, 1994.
19. B. N. Datta and Y. Saad, Arnoldi methods for large Sylvester-like matrix equations and an associated algorithm for partial spectrum assignment, *Lin. Alg. Appl.*, vol. 156, pp.225-244, 1991.
20. B. N. Datta and D. Sarkissian, Block algorithms for state estimation and functional observers, Proc. *IEEE Joint Conference on Control Applications and Computer-aided Control Systems Design*, pp. , 2000.
21. K. Datta, Y. P. Hong and R. B. Lee, Applications of linear transformations to matrix equations, *Lin. Alg. Appl.*, vol. 267, pp.221-240, 1997.
22. B.N. Datta and C. Hetti, Generalized Arnoldi methods for the Sylvester-observer equation and the multi-input pole placement problem, *Proc. 36th IEEE Conf. Dec. Contr.*, pp. 4379-4383, 1997.
23. P. Dorato, C. Abdallah and V. Cerone, *Linear Quadratic Control: An Introduction*, Prentice Hall, Englewood Cliffs, NJ, 1995.
24. J.C. Doyle and G. Stein, Multivariable feedback design: Concepts for a classical/modern synthesis, *IEEE Trans. Automat. Control*, vol. AC-26, pp. 4-16, 1981.
25. J. C. Doyle, Guaranteed margins for LQG regulators, *IEEE Trans. Automat. Control*, vol. AC-23, pp.756-757, 1978.
26. J. C. Doyle and G. Stein, Robustness with observers, *IEEE Trans. Automat. Control*, vol. AC-24, pp.607-611, 1979.
27. B. Friedland, *Control System Design*, McGraw-Hill, New York, 1986.
28. D. Gangsaas, Application of modern synthesis to aircraft control: three case studies, *IEEE Trans. Automat. Control*, vol. AC-31, pp.995-1104, 1986.
29. A. Ghavimi and A. J. Laub, Numerical methods for nearly constrained matrix Sylvester equations, *SIAM J. Matrix Anal. Appl.*, vol. 17, no. 1, pp.212-221, 1996.
30. T. Kailath, A.H. Sayed, and B. Hassibi, *Linear Estimation*, Prentice Hall, Englewood Cliffs, NJ, 2000.

31. T. Kailath, *Linear Systems*. Prentice Hall, Englewood Cliffs, NJ, 1980.
32. R. E. Kalman, Contribution to the theory of optimal control, *Bol. Soc. Matem. Mex.*, vol. 5, pp.102-119, 1960.
33. R. E. Kalman and R. S. Bucy, New results in linear filtering and prediction theory, *ASME Trans. Ser. D: J. Basic Engr.*, vol. 83, pp.95-107, 1961.
34. R.E. Kalman, When is a linear control system optimal? *ASME Trans. Ser. D: J. Basic Engr.*, vol. 86, pp. 51-60, 1964.
35. V. Kučera, *Discrete Linear Control*, John Wiley & Sons, New York, 1979.
36. H. J. Kushner, *Introduction to Stochastic Control*, Holt, Rinehart, and Winston, New York, 1971.
37. H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.
38. F.L. Lewis, *Optimal Control*, John Wiley & Sons, New York, 1986.
39. F.L. Lewis, *Applied Optimal Control and Estimation*, Prentice Hall, Englewood Cliffs, NJ., 1992.
40. W.S. Levine (Editor) *The Control Handbook*, CRC Press and IEEE Press, Boca Raton, Florida, 1996.
41. D.G. Luenberger, Observing the state of a linear system, *IEEE Trans. Mil. Electr.*, vol. 8, 74-80, 1964.
42. D.G. Luenberger, Observers for multivariable systems, *IEEE Trans. Automat. Control*, vol. 11, pp. 190-197, 1966.
43. D.G. Luenberger, An introduction to observers, *IEEE Trans. Automat. Control*, vol. AC-16, pp. 596-602, 1971.
44. D.G. Luenberger, *Introduction to Dynamic Systems; Theory, Models, and Applications*, John Wiley & Sons, New York, 1979.
45. M.S. Mahmoud, Structural properties of discrete systems with slow and fast modes, *Large Scale Systems*, vol. no. 3, pp.227-336, 1982.
46. MATLAB User's Guide, The Math Works, Inc., Natick, MA, 1992.
47. P. S. Maybeck, *Stochastic Models, Estimation, and Control*, vols. 1-3, Academic Press, New York, 1979.

48. D. McLean, *Automatic Flight Control Systems*, International Series in Systems and Control Engineering, Prentice Hall, London, 1990.
49. M. G. Safonov and M. Athans, Gain and phase margins of multiloop LQG regulators, *IEEE Trans. Automat. Control*, AC-22, pp. 173-179, 1977.
50. C. E. DeSouza and M. D. Fragoso, On the existence of maximal solution for generalized algebraic Riccati equations arising in stochastic control, *Syst. Contr. Lett.* vol. 14, pp. 233-239, (1990).
51. G. Stein and M. Athans, The LQG/LTR procedure for multi-variable feedback control design, *IEEE Trans. Automat. Control*, vol. AC-32, pp. 105-114, 1987.
52. C.-C. Tsui, An algorithm for the design of multi-functional observers, *IEEE Trans. Auto. Control*, vol. AC-30, pp. 89-93, 1985.
53. C.-C. Tsui, A new approach to robust observer design, *Int. J. Control*, vol. 47, pp. 745-751, 1988.
54. C.-C. Tsui, On the solution to matrix equation $TA - FT = LC$, *SIAM J. Matrix Anal. Appl.*, vol.14, pp. 33-44, 1993.
55. P. Van Dooren, Reduced order observers: A new algorithm and proof, *Syst. Contr. Lett.*, vol. 4, pp. 243-251, 1984.
56. K. Zhou, J.C. Doyle and K. Glover, *Robust and Optimal Control*, Prentice Hall, Upper Saddle River, NJ., 1996.

END

Chapter 13

NUMERICAL SOLUTIONS AND CONDITIONING OF ALGEBRAIC RICCATI EQUATIONS

Contents

13.1 Introduction	585
13.2 The Existence and Uniqueness of the Stabilizing Solution of the CARE	587
13.3 The Existence and Uniqueness of the Stabilizing Solution of the DARE	595
13.4 Conditioning of the Riccati Equations	596
13.4.1 Conditioning of the CARE	596
13.4.2 Conditioning of the DARE	601
13.5 Computational Methods for Riccati Equations	604
13.5.1 The Invariant Subspace Methods	605
13.5.2 The Deflating Subspace Methods	614
13.5.3 The Matrix Sign Function Methods	624
13.5.4 Newton's Methods	631
13.6 The Schur and Inverse-Free Generalized Schur Methods for the Descriptor Riccati Equations	645
13.6.1 The Generalized Schur Method for the DCARE	645
13.6.2 The Inverse-Free Generalized Schur Method for the DCARE	646
13.6.3 The Inverse-Free Generalized Schur Method for the DDARE	646
13.7 Conclusions and Table of Comparisons	647
13.8 Some Selected Software	649
13.8.1 MATLAB CONTROL SYSTEM TOOLBOX	649

13.8.2 MATCONTROL	649
13.8.3 CSP-ANM	649
13.8.4 SLICOT	650
13.8.5 MATRIX _X	650
13.9 Summary and Review	650
13.10 Chapter Notes and Further Reading	654
13.11 Introduction	671
13.12 The Existence and Uniqueness of the Stabilizing Solution of the CARE	673
13.13 The Existence and Uniqueness of the Stabilizing Solution of the DARE	681
13.14 Conditioning of the Riccati Equations	682
13.14.1 Conditioning of the CARE	682
13.14.2 Conditioning of the DARE	687
13.15 Computational Methods for Riccati Equations	690
13.15.1 The Invariant Subspace Methods	691
13.15.2 The Deflating Subspace Methods	700
13.15.3 The Matrix Sign Function Methods	710
13.15.4 Newton's Methods	717
13.16 The Schur and Inverse-Free Generalized Schur Methods for the Descriptor Riccati Equations	731
13.16.1 The Generalized Schur Method for the DCARE	731
13.16.2 The Inverse-Free Generalized Schur Method for the DCARE	732
13.16.3 The Inverse-Free Generalized Schur Method for the DDARE	732
13.17 Conclusions and Table of Comparisons	733
13.18 Some Selected Software	735
13.18.1 MATLAB CONTROL SYSTEM TOOLBOX	735
13.18.2 MATCONTROL	735
13.18.3 CSP-ANM	735
13.18.4 SLICOT	736
13.18.5 MATRIX _X	736
13.19 Summary and Review	736
13.20 Chapter Notes and Further Reading	740

Topics Covered

- Results on Existence and Uniqueness of Solutions of the CARE and DARE
- Perturbation Analyses and Condition Numbers
- The Schur Methods, Newton's Methods, and the Matrix Sign Function Methods
- Convergence Results for Newton's Methods
- The Generalized Eigenvector and the Generalized Schur Methods
- Inverse Free Generalized Schur Methods
- The Schur and Inverse-Free Schur Methods for the Descriptor Riccati Equations
- Comparative Study and Recommendations

13.1 Introduction

This chapter is devoted to the study of numerical solutions of the continuous-time algebraic Riccati equation (CARE):

$$XA + A^T X + Q - XBR^{-1}B^T X = 0 \quad (13.1.1)$$

and of its discrete counterpart (DARE)

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0. \quad (13.1.2)$$

The equation (13.1.1) is very often written in the following compact form:

$$XA + A^T X + Q - XSX = 0 \quad (13.1.3)$$

where

$$S = BR^{-1}B^T. \quad (13.1.4)$$

Similarly, in analogy with the form of the CARE in (13.1.3), the equation (13.1.2) can also be written in the compact form

$$A^T X (I + SX)^{-1} A - X + Q = 0, \quad (13.1.5)$$

where S is again given by (13.1.4).

Throughout this chapter, we assume that $R = R^T > 0$ and $Q = Q^T \geq 0$.

These equations have long been subject of research in mathematics, physics and engineering. They play major roles in many design problems in control and filter theory. As we have seen in Chapter 10, historically, AREs started as an important tool in the solution of *Linear Quadratic Optimization problems*. In recent years, they became a subject of intensive study, both from theoretical and computational viewpoints, because of their important roles in **state-space solutions of H_∞ and robust control problems**. For a brief history of the importance, applications, and historical developments of the algebraic Riccati equations, see Bittanti, et al (1991).

The following computational methods for the CARE and DARE are widely known in the literature and most of them are discussed in **Section 13.5** of this chapter.

- (i) **The Eigenvector Methods** (McFarlane (1963), Potter (1966)).
- (ii) **The Schur-Methods and the Structure-Preserving Schur Methods** (Laub (1979), Byers (1983, 1986a, 1990), Mehrmann (1988), Bunse-Gerstner and Mehrmann (1986), Benner, Mehrmann and Xu (1997)).
- (iii) **The Generalized Eigenvector and the Generalized Schur Methods** (Van Dooren (1981), Arnold and Laub (1984), Pappas, Laub and Sandell (1980), and Mehrmann (1991)).
- (iv) **The Matrix Sign Function Methods** (Roberts (1971, 1980), Bierman (1984), Byers (1987), Denman and Beavers (1976), Kenney and Laub (1995), Gardiner and Laub (1986)).
- (v) **Newton's Methods** (Kleinman (1968), Hewer (1971), Benner and Byers (1994, 1995, 1998), Guo and Lancaster (1998), Guo (1998)).

The eigenvector methods are well-known to have numerical difficulties in case the Hamiltonian matrix associated with the CARE or the symplectic matrix associated with the DARE has some multiple or near-multiple eigenvalues (the corresponding eigenvectors will be ill-conditioned). In these cases, the Schur methods, based on the real Schur decompositions of the Hamiltonian matrix for the CARE and of the symplectic matrix for the DARE, should be preferred over the eigenvector methods. The Schur method is widely used in practice for the CARE. Unfortunately, it cannot be applied to the DARE when A is singular. Indeed, even if A is theoretically nonsingular but is computationally close to a singular matrix, the Schur method for the DARE should be avoided. An alternative for the DARE then is to use the generalized Schur method which is based on the Schur decomposition of a matrix pencil and does not involve computation of the inverse of A . Having said this, it should be noted that the Schur methods and the generalized Schur methods require explicit computation of the inverse of the matrix R both for the CARE and the DARE. So, when R is close to a singular matrix, the methods of choice are the inverse-free generalized Schur methods.

Newton's methods are iterative in nature and are usually used as iterative refinement techniques for solutions obtained by the Schur methods or the matrix sign function methods. **A table of comparison of the different methods and recommendation based on this comparison are provided at the end of this chapter.**

Sections 13.2 and 13.3 deal, respectively, with the results on the **existence and uniqueness of the stabilizing solutions** of the CARE and the DARE. **The condition numbers and bounds of the condition numbers** of the CARE and DARE are identified in Section 13.4.

13.2 The Existence and Uniqueness of the Stabilizing Solution of the CARE

The goal of this section is to derive conditions under which the CARE admits a unique symmetric positive semidefinite stabilizing solution.

For this we first need to develop an important relationship between the CARE and the associated Hamiltonian matrix and some spectral properties of this matrix.

Recall from Chapter 10 that associated with the CARE is the $2n \times 2n$ Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}. \quad (13.2.1)$$

The Hamiltonian matrix H has the following interesting spectral property.

Theorem 13.2.1 *For each eigenvalue λ of H , $-\bar{\lambda}$ is also an eigenvalue of H (with the same geometric and algebraic multiplicity as λ).*

Proof: Define the $2n \times 2n$ matrix

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}, \quad (13.2.2)$$

where I is the $n \times n$ identity matrix. Then it is easy to see that $J^{-1}HJ = -JHJ = -H^T$, which shows that H and $-H^T$ are similar. Hence λ is also an eigenvalue of $-H^T$. Since the eigenvalues of $-H^T$ are the negatives of the eigenvalues of H , and the complex eigenvalues occur in conjugate pairs, the theorem is proved. ■

The following theorems show that a solution X of the CARE is determined by the associated Hamiltonian matrix.

Theorem 13.2.2 *A matrix X is a solution of the CARE if and only if the columns of $\begin{pmatrix} I \\ X \end{pmatrix}$ span an n -dimensional invariant subspace of the Hamiltonian matrix H defined by (13.2.1).*

Proof: We first prove that if the columns of $\begin{pmatrix} I \\ X \end{pmatrix}$ span an n -dimensional invariant subspace of H , then X is a solution of the CARE.

So, assume there exists an $n \times n$ matrix L such that

$$H \begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} I \\ X \end{pmatrix} L. \quad (13.2.3)$$

Multiplying both sides of (13.2.3) by J^{-1} , where J is defined by (13.2.2), we have

$$J^{-1} H \begin{pmatrix} I \\ X \end{pmatrix} = J^{-1} \begin{pmatrix} I \\ X \end{pmatrix} L \quad (13.2.4)$$

Noting that $J^{-1} = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}$, we obtain from (13.2.4)

$$\begin{pmatrix} Q & A^T \\ A & -S \end{pmatrix} \begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} -X \\ I \end{pmatrix} L \quad (13.2.5)$$

Premultiplying both sides of (13.2.5) by $[I, X]$, we get

$$XA + A^TX + Q - XSX = 0,$$

showing that X satisfies the CARE.

To prove the converse, we note that if X is a solution of the CARE, then

$$H \begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} A - SX \\ -Q - A^TX \end{pmatrix} = \begin{pmatrix} A - SX \\ X(A - SX) \end{pmatrix} = \begin{pmatrix} I \\ X \end{pmatrix} (A - SX), \quad (13.2.6)$$

that is, the columns of $\begin{pmatrix} I \\ X \end{pmatrix}$ span an invariant subspace of H .

Corollary 13.2.1 *If the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ span an n -dimensional invariant subspace of the Hamiltonian matrix H associated with the CARE and X_1 is invertible, then $X = X_2 X_1^{-1}$ is a solution of the CARE.*

Proof:

The span of the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ = the span of the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} X_1^{-1}$

= the span of the columns of $\begin{pmatrix} I \\ X_2 X_1^{-1} \end{pmatrix}$.

Therefore, by Theorem 13.2.2, we see that $X = X_2 X_1^{-1}$ is a solution of the CARE. ■

The next theorem shows how the eigenvalues of the Hamiltonian matrix H are related to those of the optimal closed-loop matrix.

Theorem 13.2.3 *Let X be a symmetric solution of the CARE. Then the eigenvalues of the Hamiltonian matrix H are the eigenvalues of $A - BK$ together with those of $-(A - BK)^T$, where $K = R^{-1}B^T X$.*

Proof: Define $T = \begin{pmatrix} I & 0 \\ X & I \end{pmatrix}$, where I and X are $n \times n$.

Then

$$\begin{aligned} T^{-1}HT &= \begin{pmatrix} I & 0 \\ -X & I \end{pmatrix} \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} \begin{pmatrix} I & 0 \\ X & I \end{pmatrix} \\ &= \begin{pmatrix} A - SX & -S \\ -(A^T X + XA + Q - XSX) & -(A - SX)^T \end{pmatrix} \\ &= \begin{pmatrix} A - SX & -S \\ 0 & -(A - SX)^T \end{pmatrix} \end{aligned} \tag{13.2.7}$$

Thus the eigenvalues of H are the eigenvalues of $A - SX$ together with those of $-(A - SX)^T$. The result now follows by noting that

$$A - SX = A - BR^{-1}B^T X = A - BK.$$

(Recall that $S = BR^{-1}B^T$). ■

Symmetric Positive Semidefinite Stabilizing Solutions of the CARE

As we have seen in Chapter 10 most applications require a symmetric positive semidefinite stabilizing solution of the associated Riccati equation. Thus, though there exist many solutions of the Riccati equations, we will be developing methods only for the symmetric positive semidefinite stabilizing solutions in this chapter.

We remind the readers that a symmetric solution X of (13.1.1) is a **stabilizing solution** if $A - BK = A - BR^{-1}B^T X = A - SX$ is stable.

For convenience of later use, we now state and prove a necessary and sufficient condition for the existence of such a solution. Proof of Theorem 13.2.4 below has been taken from Kimura (1997).

Theorem 13.2.4 (Existence and Uniqueness of the Stabilizing Solution)

Assume that $R > 0$ and $Q \geq 0, Q \neq 0$.

Then the following conditions are equivalent:

1. *The continuous-time algebraic Riccati equation*

$$XA + A^T X - XBR^{-1}B^T X + Q = 0, \quad (13.2.8)$$

has a unique symmetric positive semidefinite stabilizing solution X .

2. *(A, B) is stabilizable and the associated Hamiltonian matrix H has no pure imaginary eigenvalues.*

Proof of Necessity

First suppose that X is a stabilizing solution of the CARE. We then show that H does not have an imaginary eigenvalue.

Since X is a stabilizing solution, $A - SX$ is stable. From Theorem 13.2.3, we then have that n eigenvalues of H are stable and the other n have positive real parts. Thus H does not have a pure imaginary eigenvalue.

Proof of sufficiency

Next assume that H given in (13.2.1), with $S = BR^{-1}B^T$, has no eigenvalues on the imaginary axis. We shall then show that under the assumption of the stabilizability of (A, B) , there exists a unique stabilizing solution of the CARE.

The proof will be divided in several parts.

First of all we note that the stabilizability of (A, B) implies the stabilizability of (A, S) .

Since H has no pure imaginary eigenvalues, there are n stable eigenvalues of H (by Theorem 13.2.1).

Then

$$H \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} E \quad (13.2.9)$$

where E is a stable matrix and the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ form the eigenspace of H corresponding to these stable eigenvalues.

A. $X_2^T X_1$ is symmetric.

The relation (13.2.9) can be expressed as

$$AX_1 - SX_2 = X_1 E \quad (13.2.10)$$

and

$$-QX_1 - A^T X_2 = X_2 E. \quad (13.2.11)$$

Multiplying (13.2.10) by X_2^T on the left, we have

$$X_2^T AX_1 - X_2^T SX_2 = X_2^T X_1 E \quad (13.2.12)$$

Now taking the transpose of (13.2.11), we have

$$X_2^T A = -X_1^T Q - E^T X_2^T$$

Multiplying the last equation by X_1 to the right, we get

$$E^T X_2^T X_1 = -X_2^T A X_1 - X_1^T Q X_1 \quad (13.2.13)$$

Using (13.2.12) in (13.2.13) we then have

$$E^T X_2^T X_1 + X_2^T X_1 E = -X_2^T S X_2 - X_1^T Q X_1 \quad (13.2.14)$$

Since S and Q are symmetric, the right-hand side matrix is symmetric, and therefore the left-hand side matrix is also symmetric. This means that

$$E^T X_2^T X_1 + X_2^T X_1 E = X_1^T X_2 E + E^T X_1^T X_2$$

or

$$E^T (X_2^T X_1 - X_1^T X_2) + (X_2^T X_1 - X_1^T X_2) E = 0.$$

Since E is stable, this Lyapunov equation has a unique solution which implies that $X_2^T X_1 - X_1^T X_2 = 0$. That is $X_2^T X_1 = X_1^T X_2$, proving that $X_2^T X_1$ is symmetric.

B. X_1 is invertible.

Suppose that X_1 is not invertible. Then there exists a vector $d \neq 0$ such that

$$X_1 d = 0. \quad (13.2.15)$$

Now multiplying the transpose of (13.2.10) by d^T to the left and by $X_2 d$ to the right we have

$$\begin{aligned} d^T X_2^T S X_2 d &= -d^T E^T X_1^T X_2 d + d^T X_1^T A^T X_2 d \\ &= -d^T E^T X_2^T X_1 d + d^T X_1^T A^T X_2 d = 0, \\ &\text{(because } X_1^T X_2 = X_2^T X_1 \text{ and } X_1 d = 0\text{).} \end{aligned}$$

Again, since $S \geq 0$, we then must have

$$S X_2 d = 0.$$

The equation (13.2.10) therefore yields

$$X_1 E d = 0.$$

As this holds for all $d \in Ker(X_1)$, this means that $Ker(X_1)$ is E -invariant; that is, there exists an eigenvalue μ of E such that

$$E d' = \mu d', \quad X_1 d' = 0, \quad d' \neq 0. \quad (13.2.16)$$

Again, multiplying (13.2.11) by d' and using the relation (13.2.16) we obtain

$$(\mu I + A^T)X_2d' = 0. \quad (13.2.17)$$

Also, from (13.2.10) and (13.2.16), we have

$$SX_2d' = 0. \quad (13.2.18)$$

Since $\operatorname{Re}(\mu) < 0$ and (A, S) is stabilizable, we conclude from (13.2.18) that

$$X_2d' = 0. \quad (13.2.19)$$

Finally, $X_2d' = 0$ and $X_1d' = 0$ imply that $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ does not have the full rank which contradicts (13.2.9).

Thus X_1 is nonsingular.

C. \mathbf{X} is symmetric.

Since X_1 is nonsingular, we have from Corollary 13.2.1 that $X = X_2X_1^{-1}$ is a solution of the CARE and, since $X_2^TX_1$ is symmetric, so is X . This is seen as follows:

$$\begin{aligned} X^T - X &= X_1^{-T}X_2^T - X_2X_1^{-1} \\ &= X_1^{-T}(X_2^TX_1)X_1^{-1} - X_1^{-T}(X_1^TX_2)X_1^{-1} \\ &= X_1^{-T}(X_2^TX_1 - X_1^TX_2)X_1^{-1} = 0. \end{aligned}$$

D. \mathbf{X} is a stabilizing solution.

Multiplying (13.2.10) by X_1^{-1} to the right we obtain

$$A - SX_2X_1^{-1} = X_1EX_1^{-1}.$$

Since E is stable, so is $A - SX_2X_1^{-1} = A - SX$. Thus X is a stabilizing solution.

E. \mathbf{X} is unique.

Let X_1 and X_2 be two stabilizing solutions. Then

$$\begin{aligned} A^T X_1 + X_1 A - X_1 S X_1 + Q &= 0 \\ A^T X_2 + X_2 A - X_2 S X_2 + Q &= 0 \end{aligned}$$

Subtracting these two equations, we have

$$A^T(X_1 - X_2) + (X_1 - X_2)A + X_2 S X_2 - X_1 S X_1 = 0$$

or

$$(A - SX_1)^T(X_1 - X_2) + (X_1 - X_2)(A - SX_2) = 0$$

Since the last equation is a homogeneous Sylvester equation and the coefficient matrices $A - SX_1$ and $A - SX_2$ are both stable, it follows that $X_1 - X_2 = 0$, that is $X_1 = X_2$.

F. X is positive semidefinite.

Since X is symmetric and satisfies (13.2.8) the equation (13.2.8) can be written in the form of the following Lyapunov equation:

$$(A - BK)^T X + X(A - BK) = -Q - XSX,$$

where $K = R^{-1}B^T X$. Furthermore, $A - BK = A - BR^{-1}B^T X = A - SX$ is stable. Thus, X can be expressed in the form (see Chapter 7):

$$X = \int_0^\infty e^{(A-BK)^T t} (Q + XSX) e^{(A-BK)t} dt.$$

Since Q and S are positive semidefinite, it follows that X is positive semidefinite. ■

Theorem 13.2.5 *Let (A, B) be stabilizable and (A, Q) be detectable. Assume that $Q \geq 0$, $S \geq 0$. Then the Hamiltonian matrix*

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}$$

associated with the CARE does not have a pure imaginary eigenvalue.

Proof: The proof is by contradiction.

Suppose that H has a purely imaginary eigenvalue $j\alpha$, α is a nonnegative real number and let $\begin{pmatrix} r \\ s \end{pmatrix}$ be the corresponding eigenvector. Then

$$H \begin{pmatrix} r \\ s \end{pmatrix} = j\alpha \begin{pmatrix} r \\ s \end{pmatrix}, \quad \begin{pmatrix} r \\ s \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \quad (13.2.20)$$

Multiplying both sides of (13.2.20) by (s^*, r^*) to the left, we obtain

$$s^* Ar - r^* Qr - s^* Ss - r^* A^T s = j\alpha(s^* r + r^* s)$$

or

$$(s^* Ar - r^* A^T s) - r^* Qr - s^* Ss = j\alpha(s^* r + r^* s)$$

Considering the real part of this equation we get

$$-r^* Qr - s^* Ss = 0.$$

Since $S \geq 0$ and $Q \geq 0$ we conclude that

$$Ss = 0 \quad (13.2.21)$$

and

$$Qr = 0. \quad (13.2.22)$$

So, from (13.2.20), we have

$$Ar = j\alpha r \quad (13.2.23)$$

and

$$-A^T s = j\alpha s \quad (13.2.24)$$

Thus combining (13.2.23) and (13.2.22), we have $\begin{pmatrix} A - j\alpha I \\ Q \end{pmatrix} r = 0$.

Since (A, Q) is detectable, we have $r = 0$.

Similarly, using (13.2.24) and (13.2.21) one can show that $s = 0$.

This gives us a contradiction that $\begin{pmatrix} r \\ s \end{pmatrix}$ is an eigenvector. Thus H cannot have a pure imaginary eigenvalue.

■

An Expression for the Stabilizing Solution

Combining Theorem 13.2.4, Corollary 13.2.1 and Theorem 13.2.5, we arrive at the following result:

Theorem 13.2.6 (An Expression for the Unique Stabilizing Solution of the CARE)

Suppose that (A, B) is stabilizable and (A, Q) is detectable. Assume that $Q \geq 0$ and $R > 0$. Then there exists a unique positive semidefinite stabilizing solution X of the CARE: $XA + A^T X - X S X + Q = 0$. This solution is given by $X = X_2 X_1^{-1}$, where the columns of the matrix $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ span the invariant subspace of the Hamiltonian matrix (13.2.1) associated with its stable eigenvalues.

■

Remark: The following simple example shows that the detectability of (A, Q) is not necessary for the existence of a symmetric positive semidefinite stabilizing solution of the CARE.

$$A = \begin{pmatrix} -1 & 0 \\ 0 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad Q = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad R = 1.$$

Then (A, B) is stabilizable, but (A, Q) is not detectable. The matrix $X = \begin{pmatrix} 0 & 0 \\ 0 & 4 \end{pmatrix}$ is the stabilizing solution of the CARE and is positive semidefinite.

13.3 The Existence and Uniqueness of the Stabilizing Solution of the DARE

The existence and uniqueness of the stabilizing solution of the DARE can be studied via a symplectic matrix which takes the role of the Hamiltonian matrix of the CARE.

Definition 13.3.1 A matrix M is **symplectic** if

$$J^{-1}M^TJ = J^TM^TJ = M^{-1},$$

where J is defined by (13.2.2).

Assume A is invertible and consider the matrix

$$M = \begin{pmatrix} A + S(A^{-1})^TQ & -S(A^{-1})^T \\ -(A^{-1})^TQ & (A^{-1})^T \end{pmatrix}, \quad (13.3.1)$$

where $S = BR^{-1}B^T$, $Q = Q^T$, and $S = S^T$.

Then, it can be shown [Exercise 3] that

1. M is symplectic.
2. If λ is a nonzero eigenvalue of M , so is $\frac{1}{\lambda}$.

We now state the discrete counterparts of Theorem 13.2.5 and 13.2.6. The proofs can be found in Lancaster and Rodman (1995).

Theorem 13.3.1 Let (A, B) be discrete-stabilizable and let (A, Q) be discrete-detectable. Assume that $Q \geq 0$ and $S \geq 0$. Then the symplectic matrix (13.3.1) has no eigenvalues on the unit circle.

Suppose that the symplectic matrix M has no eigenvalues on the unit circle. Then it must have n eigenvalues inside the unit circle and n outside it. As in the continuous-time case, it then can be shown that if the columns of the matrix $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ form a basis for the invariant subspace associated with the eigenvalues inside the unit circle, then X_1 is nonsingular and $X = X_2X_1^{-1}$ is a unique symmetric positive semidefinite stabilizing solution of the DARE.

Thus, we have the following theorem as the discrete counterpart of Theorem 13.2.6.

Theorem 13.3.2 (An Expression for the Unique Stabilizing Solution of the DARE) Suppose that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Assume that $Q \geq 0, R > 0$. Then the DARE

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0$$

has a unique symmetric positive semidefinite discrete-stabilizing solution X .

Furthermore X is given by $X = X_2X_1^{-1}$, where the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ span the n -dimensional invariant subspace of the symplectic matrix M associated with the eigenvalues inside the unit circle.

13.4 Conditioning of the Riccati Equations

Before we describe the solution methods for Riccati equations, we state some results on the perturbation theory of such equations that will help us identify the *condition numbers* of the equations. These condition numbers, as usual, will help us understand the sensitivity of the solutions of the Riccati equations when the entries of the data matrices are slightly perturbed.

13.4.1 Conditioning of the CARE

Consider first the CARE:

$$A^T X + X A + Q - X S X = 0, \quad (13.4.1)$$

where

$$S = B R^{-1} B^T. \quad (13.4.2)$$

Let ΔA , ΔX , ΔQ , and ΔS be small perturbations in A , X , Q and S , respectively. Suppose that X is the unique stabilizing solution of the CARE and that $X + \Delta X$ is the unique stabilizing solution of the perturbed Riccati equation

$$(A + \Delta A)^T (X + \Delta X) + (X + \Delta X)(A + \Delta A) + (Q + \Delta Q) - (X + \Delta X)(S + \Delta S)(X + \Delta X) = 0. \quad (13.4.3)$$

We are interested in finding an upper bound for the relative error $\frac{\|\Delta X\|}{\|X\|}$.

Several results exist in literature. Byers (1985) and Kenney and Hewer (1990) obtained the first-order perturbation bounds and Chen (1988), Konstantinov, Petkov, and Christov (1990) gave global perturbation bound. Xu (1996) has improved Chen's result and Konstantinov, Petkov, Gu, and Postlethwaite (1995) have sharpened the results of Konstantinov, Petkov, and Christov (1990). The most recent result in this area is due to Sun (1998), who has improved Xu's result. We present below Sun's result and the condition numbers derived on the basis of this result.

Following the notations in Byers (1985), we define three operators:

$$\Omega(Z) = (A - SX)^T Z + Z(A - SX), \quad (13.4.4)$$

$$\Theta(Z) = \Omega^{-1}(Z^T X + X Z) \quad (13.4.5)$$

$$\Pi(Z) = \Omega^{-1}(X ZX). \quad (13.4.6)$$

Note: Since $A_C = A - SX$ is stable, Ω^{-1} exists. In fact, if $\Omega(Z) = W$, then

$$Z = \Omega^{-1}(W) = - \int_0^\infty e^{A_C^T t} W e^{A_C t} dt,$$

and $\|\Omega^{-1}\|_F = \frac{1}{\text{sep}(A_C^T, -A_C)}.$

Define $l = \|\Omega^{-1}\|^{-1}$, $p = \|\Theta\|$, and $q = \|\Pi\|$, where $\|\cdot\|$ is any unitarily invariant norm. Then the following perturbation result due to Sun (1998) holds:

Theorem 13.4.1 (A Perturbation Bound for the Continuous-time Algebraic Riccati Equation)

Let X and $X + \Delta X$ be, respectively, the symmetric positive semidefinite stabilizing solutions of the CARE (13.4.1) and the perturbed CARE (13.4.3).

Then, for sufficiently small $[\Delta Q, \Delta A, \Delta S]$,

$$\frac{\|\Delta X\|}{\|X\|} \lesssim \frac{\|Q\|}{l\|X\|} \cdot \frac{\|\Delta Q\|}{\|Q\|} + p \frac{\|A\|}{\|X\|} \cdot \frac{\|\Delta A\|}{\|A\|} + q \frac{\|S\|}{\|X\|} \cdot \frac{\|\Delta S\|}{\|S\|}. \quad (13.4.7)$$

■

Using the results of Theorem 13.4.1, Sun has defined a set of condition numbers of the CARE. The numbers

$$\kappa_{CARE}^{AB}(Q) = \frac{1}{l}, \quad \kappa_{CARE}^{AB}(A) = p, \quad \text{and} \quad \kappa_{CARE}^{AB}(S) = q$$

are the **absolute condition numbers** with respect to Q, A, S , respectively.

The numbers $\kappa_{CARE}^{REL}(Q) = \frac{\|Q\|}{l\|X\|}$, $\kappa_{CARE}^{REL}(A) = \frac{p\|A\|}{\|X\|}$ and $\kappa_{CARE}^{REL}(S) = \frac{q\|S\|}{\|X\|}$ are then the **relative condition numbers**.

Moreover, the scalar

$$\kappa_{CARE}^{REL}(X) = \frac{1}{\|X\|} \sqrt{\left(\frac{\|Q\|}{l}\right)^2 + (p\|A\|)^2 + (q\|S\|)^2} \quad (13.4.8)$$

can be regarded as the **relative condition number of X** .

Using a local linear estimate, Byers (1985) has obtained an approximate condition number, given by

$$\kappa_{CARE}^B = \frac{1}{\|X\|_F} \left(\frac{\|Q\|_F}{l} + p\|A\|_F + q\|S\|_F \right),$$

in which the operator norm $\|\cdot\|$ for defining l, p , and q is induced by the Frobenius norm $\|\cdot\|_F$. The above is known as **Byers' approximate condition number**. Indeed, taking the Frobenius norm in (13.4.8), it is easy to see that

Theorem 13.4.2

$$\frac{1}{\sqrt{3}} \kappa_{CARE}^B \leq \kappa_{CARE}^{REL}(X) \leq \kappa_{CARE}^B$$

Expressions for l, p , and q

If the operator norm $\|\cdot\|$ for defining l, p , and q is induced by the Frobenius norm $\|\cdot\|_F$, then it can be shown Sun (1998) that

$$l = \|T^{-1}\|_2^{-1}, \quad p = \|T^{-1}(I_n \otimes X + (X^T \otimes I_n)E)\|_2$$

and

$$q = \|T^{-1}(X^T \otimes X)\|_2,$$

where

$$T = I_n \otimes (A - SX)^T + (A - SX)^T \otimes I_n,$$

and E is the vec-permutation matrix:

$$E = \sum_{i,j=1}^n (e_i e_j^T) \otimes (e_j e_i^T).$$

Remarks. A recent paper of Petkov, Konstantinov and Mehrmann (1998) contains results on estimating the quantities l, p and q .

Bounds on Condition Numbers Using Lyapunov Equations

Computing the quantities l, p , and q using the Kronecker products is computationally intensive. On the other hand, Kenney and Hewer (1990) have obtained an upper and a lower bound of κ_{CARE}^B using solutions of certain Lyapunov equations, which are certainly computationally much less demanding than computing Kronecker products. Using these results, ill-conditioning of κ_{CARE} can be more easily detected.

Assume that $A - SX$ is stable and let H_k be the solution to the Lyapunov equation:

$$(A - SX)^T H_k + H_k (A - SX) = -X^k, \quad k = 0, 1, 2. \quad (13.4.9)$$

Furthermore, let's define $H_1^{(1)}$ as follows:

Set $\tilde{Q} = 2X$, and solve the successive Lyapunov equations:

$$(A - SX)^T \tilde{H} + \tilde{H} (A - SX) = \tilde{Q}. \quad (13.4.10)$$

and

$$(A - SX)H + H(A - SX)^T = \tilde{H}. \quad (13.4.11)$$

Let $W = 2XH$ and $H_1^{(1)} = \Theta\left(\frac{W}{\|W\|}\right)$.

Define

$$U = \frac{\|H_0\| \|Q\| + 2\|H_0\|^{\frac{1}{2}} \|H_2\|^{\frac{1}{2}} \|A\| + \|H_2\| \|S\|}{\|X\|} \quad (13.4.12)$$

$$L = \frac{\|H_0\| \|Q\| + \|H_1^{(1)}\| \|A\| + \|H_2\| \|S\|}{\|X\|} \quad (13.4.13)$$

Then it can be shown that

$$L \leq \kappa_{CARE}^B \leq U \quad (13.4.14)$$

From the relations (13.4.12)-(13.4.14), we see that κ_{CARE}^B will be large (and consequently the CARE will be ill-conditioned) if H_0 , $H_1^{(1)}$, and H_2 have large norms (relative to that of X). Conversely, if the norms of H_0 , $H_1^{(1)}$, and H_2 are not large, then the CARE will be well-conditioned.

If the norms vary widely in the sense that there is a mixture of large and small norms, then there will be **selective sensitivity**. More specifically, the ratios $r_1 = \frac{\|H_0\| \|Q\|}{\|X\|}$,

$r_2 = \frac{\|H_1^{(1)}\| \|A\|}{\|X\|}$, and $r_3 = \frac{\|H_2\| \|S\|}{\|X\|}$ measure, respectively, the sensitivity of X with respect to perturbations in the matrix Q , the matrix A , and the matrix S .

Example 13.4.1 (An Ill-Conditioned CARE)

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0.0010 & 4 & 5 \\ 0 & 7 & 8 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad R = 1$$

$$Q = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 5 & 3 \\ 1 & 3 & 5 \end{pmatrix}.$$

$$X = 10^9 \begin{pmatrix} 0 & 0.0003 & 0.0004 \\ 0.0003 & 4.5689 & 5.3815 \\ 0.0004 & 5.3815 & 6.3387 \end{pmatrix}.$$

The residual norm of the solution X : $\|XA + A^TX - XSX + Q\| = O(10^{-5})$.

$$\begin{aligned} \|H_0\| &= 5.6491 \times 10^8 \\ \|H_1\| &= 1.8085 \times 10^9 \\ \|H_2\| &= 4.8581 \times 10^{18} \end{aligned}$$

U and L are both of order 10^8 .

Thus the Riccati equation is expected to be ill-conditioned with the given data.

Indeed, this is an example of mixed sensitivity. Note that the ratios r_2 and r_3 are large, but r_1 is quite small. Thus, X should be sensitive with respect to perturbation in A and S . This is verified as follows.

Let M_{new} stand for a perturbed version of the matrix M and X_{new} stands for the new solution of the ARE with the perturbed data.

Case 1. Perturbation in A. Let A be perturbed to $A + \Delta A$, where

$$\Delta A = 10^{-8} \begin{pmatrix} 3.169 & 2.668 & 3.044 \\ -1.259 & -0.5211 & -2.364 \\ 2.798 & 3.791 & -3.179 \end{pmatrix}$$

The matrices B, Q , and R remain unperturbed.

Then

$$\text{Relative error in } X : \frac{\| X_{new} - X \|}{\| X \|} = 4.074 \times 10^{-5}$$

$$\text{Relative perturbation in } A : \frac{\| A_{new} - A \|}{\| A \|} = 4.916 \times 10^{-9}.$$

Case 2. Perturbation in B. Let B be perturbed to $B + \Delta B$,

where

$$\Delta B = 10^{-8} \begin{pmatrix} -4.939 \\ 0.7715 \\ -0.9411 \end{pmatrix}$$

A, Q, R remain unperturbed.

$$\text{Relative error in } X : \frac{\| X_{new} - X \|}{\| X \|} = 7.589 \times 10^{-5}$$

$$\text{Relative perturbation in } B : \frac{\| B_{new} - B \|}{\| B \|} = 5.086 \times 10^{-8}$$

Case 3. Perturbation in Q. The matrix Q is perturbed such that the relative error in $Q : \frac{\| Q_{new} - Q \|}{\| Q \|} = 4.048 \times 10^{-9}$. The matrices A, B , and R remain unperturbed.

$$\text{Then the relative error in } X : \frac{\| X_{new} - X \|}{\| X \|} = 4.048 \times 10^{-9}.$$

Note: All the solutions to the CARE in this example were computed using the Schur method (**Algorithm 13.5.1**) followed by Newton's iterative refinement procedure (**Algorithm 13.5.8**). The residual norms of the solutions obtained by the Schur method alone were of order 10^5 . On the other hand, the residual norm of the solution with the Schur method followed by Newton's iterative procedure was, in each case, of order 10^{-5} .

Example 13.4.2 (A Well-Conditioned CARE).

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{and } R = 1.$$

In this case $\| H_0 \| = 0.3247$, $\| H_1 \| = 0.1251$, $\| H_2 \| = 0.0510$ and $U = 3.1095$

The CARE is, therefore, expected to be well-conditioned.

Indeed, if (1, 1) entry of A is perturbed to -0.9999999 , and the other data remain unchanged, then we find

$$\text{Relative Error in } X: \frac{\| X_{new} - X \|}{\| X \|} = 5.6482 \times 10^{-8}$$

Relative perturbation in A : $\frac{\| A_{new} - A \|}{\| A \|} = 3.1097 \times 10^{-8}$, where A_{new} and X_{new} , respectively, denote the perturbed A and the solution of the CARE with the perturbed data.

Conditioning and Accuracy

Suppose that \hat{X} is an approximate stabilizing solution of the CARE

$$XA + A^T X - XSX + Q = 0,$$

where $S = BR^{-1}B^T$ and let $Res(\hat{X}) = \hat{X}A + A^T\hat{X} - \hat{X}S\hat{X} + Q$.

Then the question arises: If **$Res(\hat{X})$ is small, does it guarantee that the error in the solution is also small [Exercise 9]**. Recall from Chapter 3 that in the case of linear system problem, it has been shown that the smallness of the residual does not guarantee that the error in the solution is small, if the linear system problem is ill-conditioned. Similar result can be proved in the case of the Riccati equations (see Kenney, Laub, and Wette (1990)). **The result basically says that even if the residual is small, the computed solution may be inaccurate, if the CARE is ill-conditioned.** On the other hand, if $Res(\hat{X})$ is small and the CARE is well-conditioned, then the solution is guaranteed to be accurate. Below, we quote a recent result of Sun (1997a) which is an improvement of the result of Kenney, Laub and Wette (1990).

Theorem 13.4.3 (Residual Bound of an Approximate Stabilizing Solution). Let $\hat{X} \geq 0$ approximate the positive semidefinite stabilizing solution X to the CARE. Define the linear operator $T : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times n}$ by

$$T(Z) = (A - S\hat{X})^T Z + Z(A - S\hat{X}), \quad Z = Z^T \in \mathbb{R}^{n \times n}$$

Assume that $4 \| T^{-1} \| \| T^{-1}(Res(\hat{X})) \| \| S \| < 1$ for any unitarily invariant norm $\| \cdot \|$, then

$$\frac{\| \hat{X} - X \|}{\| \hat{X} \|} \leq \frac{2}{1 + \sqrt{1 - 4 \| T^{-1} \| \| T^{-1}Res(\hat{X}) \| \| S \|}} \frac{\| T^{-1}Res(\hat{X}) \|}{\| \hat{X} \|}.$$

13.4.2 Conditioning of the DARE

Consider now the DARE

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$

The condition number of the DARE, denoted by κ_{DARE} , may be obtained by means of the **Frechet derivative** of the DARE (Gudmundsson, Kenney, and Laub (1992)).

$$\text{Define } A_d = A - B(R + B^T X B)^{-1} B^T X A, \quad S = B R^{-1} B^T. \quad (13.4.15)$$

Assume that X is the stabilizing solution of the DARE. Then the condition number of the DARE is given by:

$$\kappa_{DARE} = \frac{\| [Z_1, Z_2, Z_3] \|_2}{\| X \|_F} \quad (13.4.16)$$

where

$$Z_1 = \| A \|_F P^{-1} (I \otimes A_d^T X + (A_d^T X \otimes I) E), \quad (13.4.17)$$

$$Z_2 = - \| S \|_F P^{-1} (A^T X (I + S X)^{-1} \otimes A^T X (I + S X)^{-1}) \quad (13.4.18)$$

and

$$Z_3 = \| Q \|_F P^{-1}. \quad (13.4.19)$$

In the above, E is the vec-permutation matrix:

$$E = \sum_{i,j=1}^n e_i e_j^T \otimes e_j e_i^T, \quad (13.4.20)$$

and P is a matrix representation of the Stein operator

$$\Omega(Z) = Z - A_d^T Z A_d. \quad (13.4.21)$$

Note that, since A_d is discrete-stable, P^{-1} exists.

The condition number (13.4.16) measures the sensitivity of the stabilizing solution X of the DARE with respect to first-order perturbations.

Assume that the bounds for ΔA , ΔS , and ΔQ are **sufficiently small**. Then, using first-order perturbation only, it can be shown [**Exercise 7**] that the following quantity is an **approximate condition number** of the DARE

$$\frac{2\|A\|_F^2 \frac{\|Q\|_F}{\|X\|_F} + \|A\|_F^2 \|S\|_F \|X\|_F}{sep_d(A_d^T, A_d)}, \quad (13.4.22)$$

where

$$sep_d(A_d^T, A_d) = \min_{X \neq 0} \frac{\| A_d^T X A_d - X \|_F}{\| X \|_F}. \quad (13.4.23)$$

Note: The quantity $sep(A_d^T, A_d)$ can be computed as the minimum singular value of the matrix

$$A_d^T \otimes A_d^T - I_{n^2}.$$

Remark: A perturbation theorem for the DARE, analogous to Theorem 13.4.1 (for the CARE), and the absolute and relative condition numbers using the results of that theorem, can be obtained. For details see Sun (1998).

Also, a recent paper of Sima, Petkov and Van Huffel (2000) contains efficient and reliable condition number estimators both for the CARE and DARE.

Example 13.4.3 (An Ill-Conditioned DARE). Let's take A, B, Q , and R the same as in Example 13.4.1.

$$\text{Let } A_{\text{new}} = \begin{pmatrix} 0.9999999 & 2 & 3 \\ 0.0010 & 4 & 5 \\ 10^{-8} & 7 & 8 \end{pmatrix}.$$

Let B, Q and R remain unchanged.

The solution X of the DARE (computed by MATLAB function **dare**) is

$$X = 10^{10} \begin{pmatrix} 0.0000 & 0.0005 & 0.0005 \\ 0.0005 & 5.4866 & 6.4624 \\ 0.0005 & 6.4624 & 7.6118 \end{pmatrix}$$

The solution X_{new} of the perturbed version of the DARE is

$$X_{\text{new}} = 10^{10} \begin{pmatrix} 0.0000 & 0.0005 & 0.0005 \\ 0.0005 & 5.4868 & 6.4622 \\ 0.0005 & 6.4622 & 7.6116 \end{pmatrix}$$

Relative error in X : $\frac{\|X - X_{\text{new}}\|}{\|X\|} = 2.3535 \times 10^{-5}$, while the perturbations in A , were of order 10^{-9} .

Example 13.4.4 (A Well-conditioned DARE). Let $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3.999 & 6 & 7 \end{pmatrix}$.

Take B, Q , and R the same as in Example 13.4.1

Let

$$A_{\text{new}} = \begin{pmatrix} 0.9990 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 6 & 7 \end{pmatrix},$$

$B_{\text{new}} = B$, $Q_{\text{new}} = Q$, and $R_{\text{new}} = R$.

Then both the relative error in X and the relative perturbation in A are of $O(10^{-4})$. In this case, $\text{sep}(A_d^T, A_d) = 0.0011$.

13.5 Computational Methods for Riccati Equations

The existing computational methods (**listed in the Introduction**) for the algebraic Riccati equations can be broadly classified into four classes:

- (I) The Invariant Subspace Methods
- (II) The Deflating Subspace Methods
- (III) The Matrix Sign-Function Methods
- (IV) Newton's Methods

The eigenvector and the Schur vector methods are examples of the invariant subspace methods. The generalized eigenvector and the generalized Schur vector methods are examples of the deflating subspace methods.

The following methods have been included in our discussions here.

For the CARE:

- **The eigenvector method** (Section 13.5.1)
- **The Schur method** (Algorithm 13.5.1)
- **The Hamiltonian Schur method** (Section 13.5.1)
- **The inverse-free generalized Schur method** (Algorithm 13.5.3)
- **The matrix sign-function method** (Algorithm 13.5.6)
- **Newton's method** (Algorithm 13.5.8)
- **Newton's method with line search** (Algorithm 13.5.9)

For the DARE:

- **The Schur method** (Section 13.5.1)
- **The generalized Schur method** (Algorithm 13.5.2)
- **The inverse-free generalized Schur method** (Algorithm 13.5.4).
- **The matrix sign function method** (Algorithm 13.5.7)
- **Newton's method** (Algorithm 13.5.10)
- **Newton's method with line search** (Algorithm 13.5.11)

13.5.1 The Invariant Subspace Methods

The invariant subspace methods for the CARE and DARE are, respectively, based on Theorems 13.2.6 and 13.3.2.

Theorem 13.2.6 (Theorem 13.3.2) tells us that a unique stabilizing solution of the CARE (DARE) can be computed by constructing a basis for the stable invariant subspace of the associated Hamiltonian matrix H (the symplectic matrix M). Such bases can be constructed using either the eigendecomposition or the real Schur decomposition of the matrices H and M , giving rise, respectively, to the eigenvector and the Schur vector methods. **We remind the readers that we make the assumption $Q = Q^T \geq 0$ and $R = R^T > 0$ throughout the whole section.**

The Eigenvector Method for the CARE

Assume that (A, B) is stabilizable and (A, Q) is detectable.

Then the Hamiltonian matrix H does not have a purely imaginary eigenvalue. Let H be diagonalizable and have the eigendecomposition:

$$V^{-1}HV = \begin{pmatrix} -\bar{\Lambda} & 0 \\ 0 & \Lambda \end{pmatrix},$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ and $\lambda_1, \dots, \lambda_n$ are the n eigenvalues of H with positive real parts. Let V be partitioned conformably:

$$V = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}$$

such that $\begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix}$ is the matrix of eigenvectors corresponding to the stable eigenvalues. Then it is easy to see that

$$H \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} (-\bar{\Lambda}).$$

From Theorem 13.2.6, we then have that $X = V_{21}V_{11}^{-1}$ is a unique positive semidefinite stabilizing solution.

Remarks. The eigenvector method, in general, cannot be recommended for practical use. The method becomes highly unstable if the Hamiltonian matrix H is defective or nearly defective; that is, if there are some multiple or near multiple eigenvalues of H . In these cases, the matrix V_{11} will be poorly conditioned, making $X = V_{21}V_{11}^{-1}$ inaccurate; and this might happen even if the CARE itself is not ill-conditioned.

The eigenvector method, in principle, is applicable even when H is not diagonalizable by computing the principal vectors, but again is not recommended in practice.

MATCONTROL Note: The eigenvector method for the CARE has been implemented in MATCONTROL function **riceigc**.

The Eigenvector Method for the DARE

An analogous method for the DARE can be developed by taking the eigendecomposition of the associated symplectic matrix M . However, since forming the matrix M requires computation of A^{-1} , **the eigenvector method for the DARE works only when A is nonsingular.** **But even in this case, the results will be inaccurate if A is ill-conditioned.** Moreover, the method will have the same sort of difficulties as those mentioned above for the CARE. We, thus, skip the description of the eigenvector method for the DARE.

The Schur Vector Method for the CARE

The numerical difficulties of the eigenvector method for the CARE may somehow be reduced or eliminated if the Hamiltonian matrix H is transformed to **an ordered Real Schur form** by using the QR iteration algorithm, rather than using its eigendecomposition.

Let $U^T H U$ be an **ordered Real Schur matrix**:

$$U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix},$$

where the eigenvalues of H with negative real parts have been stacked in T_{11} and those with positive real parts are stacked in T_{22} (Note that because of our assumptions that (A, B) is stabilizable and (A, Q) is detectable, H does not have a purely imaginary eigenvalue (**Theorem 13.2.5**)).

Let $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$ be a conformable partitioning of U . Then

$$H \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} T_{11}$$

By Theorem 13.2.6, the matrix $X = U_{21} U_{11}^{-1}$ is then the unique positive semidefinite stabilizing solution of the CARE.

The above discussion leads to the following algorithm, called the *Schur algorithm*, due to Laub (1979).

Algorithm 13.5.1 The Schur Algorithm for the CARE

Inputs: A - The $n \times n$ state matrix
 B - The $n \times m$ input ($m \leq n$) matrix
 Q - The $n \times n$ state weighting matrix
 R - The $m \times m$ control weighting matrix

Output: X - The unique symmetric positive semidefinite stabilizing solution of the CARE.

Assumptions:

1. (A, B) is stabilizable and (A, Q) is detectable.
2. $Q \geq 0, R > 0$.

Step 1. Form the Hamiltonian matrix

$$H = \begin{pmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{pmatrix}.$$

Step 2. Transform H to the **ordered real Schur form**:

$$U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix},$$

where the n eigenvalues of H with negative real parts are contained in T_{11} .

Step 3. Partition U conformably:

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}.$$

Step 4. Compute the solution X by solving the linear systems:

$$XU_{11} = U_{21}.$$

■

Software for the ordered real Schur form:

The ordered real Schur form of H can be obtained by transforming H first to the real Schur form (RSF) by orthogonal similarity, followed by another orthogonal similarity applied to the RSF to achieve the desired ordering of the eigenvalues (**See Chapter 4**).

There exists an efficient algorithm and an associated software developed by Stewart (1976) for this purpose: Algorithm 506 of the Association for Computing Machinery Trans. Math Software (1976), 275-80. See also the LAPACK routine STRSEN.

The MATLAB program **orderschur** from MATCONTROL can also be used for this purpose.

Flop-count. The Schur-method is based on reduction to real Schur form, which is done by QR iterations algorithm; so, an exact flop-count cannot be given. However, assuming that the average number of iterations per eigenvalue is 2, about $234n^3$ flops will be necessary to execute the algorithm, which can be seen as follows:

- 1 Reduction to the Real-Schur Form (including the formation of the matrix U): About $26(2n)^3 = 208n^3$.
2. Solution of $XU_{11} = U_{21}$: $\frac{8}{3}n^3$
3. Ordering the real Schur form: At most $24n^3$

Total: (About) $234n^3$ (These counts are very approximate).

Example 13.5.1 Consider solving the CARE with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad Q = I_{3 \times 3}, \quad S = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

$$B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad R = 1.$$

Step 1. Form the Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} = \left(\begin{array}{ccc|ccc} -1 & 1 & 1 & -1 & -1 & -1 \\ 0 & -2 & 0 & -1 & -1 & -1 \\ 0 & 0 & -3 & -1 & -1 & -1 \\ \hline -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 & 3 \end{array} \right)$$

Step 2. Transform the Hamiltonian matrix to the ordered real Schur form

$$U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

$$= \left(\begin{array}{ccc|ccc} -2.9940 & -0.0216 & 1.3275 & & & * \\ 0 & -2.1867 & 0.7312 & & & \\ 0 & -0.2573 & -1.9055 & & & \\ \hline & 0 & & 2.9940 & 1.3285 & 0.2134 \\ & & & 0 & 1.9623 & 0.2434 \\ & & & 0 & -0.7207 & 2.1298 \end{array} \right)$$

The eigenvalues of T_{11} are: $-2.9940, -2.0461 + 0.4104j, -2.0461 - 0.4104j$. Thus, all the stable eigenvalues are contained in T_{11} .

Step 3. Extract U_{11} and U_{21} from U :

$$U_{11} = \begin{pmatrix} 0.4417 & 0.3716 & 0.7350 \\ 0.0053 & -0.8829 & 0.3951 \\ -0.8807 & 0.1802 & 0.3986 \end{pmatrix}$$

$$U_{21} = \begin{pmatrix} 0.1106 & 0.0895 & 0.3260 \\ 0.0232 & -0.1992 & 0.1552 \\ -0.1285 & 0.0466 & 0.1199 \end{pmatrix}$$

Step 4. Compute the stabilizing solution

$$X = U_{21}U_{11}^{-1} = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}.$$

The eigenvalues of $A - SX$ are: $-2.0461 + 0.4104j, -2.0461 - 0.4104j, -2.9940$. Thus, $A - SX$ is stable; that is, X is a unique stabilizing solution.

MATCONTROL Note. The Schur method for the CARE (using ordered real Schur form) has been implemented in MATCONTROL function **ricschc**.

Stability Analysis of the Schur Method and Scaling

The round-off properties of the Schur method are quite involved. It can be shown (Petkov, et al (1991)) that the relative error in the computed solution is proportional to $\|U_{11}^{-1}\| / \text{sep}(T_{11}, T_{22})$. This means that the Schur method can be numerically unstable even if the CARE is not ill-conditioned.

However, the difficulty can be overcome by proper scaling (Kenney, Laub and Wette (1989)).

Thus, for all practical purposes, the Schur method, when combined with an appropriate scaling, is numerically stable. For a discussion on scaling procedure, see Kenney, Laub, and Wette (1989), and Benner (1997). See also Pandey (1993).

Benner (1997) has given an extensive discussion on scaling. Based on several existing scaling strategies and considering the practical difficulties with these strategies, he has proposed a mixture of these procedures for scaling the CARE. Benner's strategy is as follows:

Write the CARE

$$XA + A^T X - XSX + Q = 0$$

in the form:

$$X_\rho A_\rho + A_\rho^T X_\rho - X_\rho S_\rho X_\rho + Q = 0$$

where $A_\rho = \rho A$, $A_\rho^T = (\rho A)^T$, $X_\rho = \frac{X}{\rho}$, and $S_\rho = \rho^2 S$, ρ being a positive scalar.

Choose ρ as

$$\rho = \begin{cases} \frac{\|S\|_2}{\|Q\|_2} & \text{if } \|Q\|_2 > \|S\|_2 \\ \frac{\|A\|_2}{\|S\|_2} & \text{if } \|Q\|_2 \leq \|S\|_2 \text{ and } \|Q\|_2 \|S\|_2 < \|A\|_2^2 \\ 1 & \text{otherwise.} \end{cases}$$

For a rationale of choosing ρ this way, see Benner (1997).

Note: Note that the relative condition number of the CARE remains invariant under the above scaling.

The Schur Method for the DARE

The Schur method for the DARE

$$A^T X A - X - A^T X B (R + B^T X B)^{-1} B^T X A + Q = 0$$

is analogous. Form the symplectic matrix

$$M = \begin{pmatrix} A + S(A^{-1})^T Q & -S(A^{-1})^T \\ -(A^{-1})^T Q & (A^{-1})^T \end{pmatrix},$$

where $S = BR^{-1}B^T$.

For a unique stabilizing solution of the DARE, we assume that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Then by Theorem 13.3.1, the symplectic matrix M does not have an eigenvalue on the unit circle. In this case M can be transformed to an ordered real Schur form such that the eigenvalues with moduli less than 1 appear in the first block; that is, an orthogonal matrix U can be constructed such that $U^T M U = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix}$, where each eigenvalue of S_{11} is inside the unit circle. Partition U conformably: $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$. Then by Theorem 13.3.2, $X = U_{21} U_{11}^{-1}$ is a unique positive semidefinite stabilizing solution of the DARE.

Remarks: (1) Since one needs to form A^{-1} explicitly to compute M , the Schur method for the DARE is not applicable if A is singular. Even if A is theoretically nonsingular, **the method is expected to give an inaccurate answer in case A is ill-conditioned with respect to inversion.**

(2) A slightly faster method (Sima (1996), p. 244) forms the matrix M^{-1} and orders the Real Schur form so that the eigenvalues with modulii less than 1 appear in the first block.

MATCONTROL Note. The Schur method for the DARE has been implemented in MATCONTROL function **ricschd**.

The Hamiltonian-Schur Methods for the CARE

The Schur methods for the algebraic Riccati equations are based on orthogonal similarity transformations of the associated Hamiltonian and symplectic matrices to real Schur forms. The rich structures of these matrices are, however, not exploited in these methods. The Hamiltonian and the symplectic matrices are treated just as $2n \times 2n$ general matrices in these methods. **It would be useful if methods could be developed that could take advantage of Hamiltonian and Symplectic structures.** Such structure-preserving methods, besides reflecting physical structures, are often faster.

Theorem 13.5.1 below shows that developments of such structure-preserving methods are possible.

Definition 13.5.1 If a matrix U is both symplectic and unitary, it is called a **symplectic-unitary matrix**. A **symplectic-orthogonal** matrix can similarly be defined.

From the above definition, it follows that an $2n \times 2n$ symplectic-unitary matrix U can be written as

$$U = \begin{pmatrix} U_{11} & U_{12} \\ -U_{12} & U_{11} \end{pmatrix},$$

where U_{11} and U_{12} are $n \times n$. If \hat{U} is $n \times n$ unitary, then

$$U = \begin{pmatrix} \hat{U} & 0_{n \times n} \\ 0_{n \times n} & \hat{U} \end{pmatrix}$$

is symplectic-unitary.

Theorem 13.5.1 (The Hamiltonian-Schur Decomposition (HSD) Theorem) (Paige and Van Loan (1981)). If the real parts of all the eigenvalues of a Hamiltonian matrix H are nonzero, then there exists a symplectic-orthogonal matrix U and a Hamiltonian matrix T such that

$$U^T H U = T = \begin{pmatrix} T_1 & T_2 \\ 0_{n \times n} & -T_1^T \end{pmatrix},$$

where T_1 is $n \times n$ upper triangular, and T_2 is $n \times n$ symmetric. Furthermore, U can be chosen so that the eigenvalues of T_1 have negative real parts.

Definition 13.5.2 The Hamiltonian matrix T in Theorem 13.5.1 is called a **Hamiltonian-Schur matrix** and the decomposition itself is called **Hamiltonian-Schur Decomposition**.

Note. The first n columns of U in the above Hamiltonian-Schur decomposition span the invariant subspace corresponding to the stabilizing solution of the CARE.

Symplectic-Schur Decomposition (SSD)

For a symplectic matrix, we have the following Theorem.

Theorem 13.5.2 (The Symplectic-Schur Decomposition (SSD) Theorem). If M is symplectic and has no eigenvalues on the unit circle, then there exists a symplectic-orthogonal matrix U such that

$$U^T M U = R = \begin{pmatrix} R_1 & R_2 \\ 0_{n \times n} & R_1^{-T} \end{pmatrix}$$

where R_1 is $n \times n$ upper triangular. Moreover, $R_2 R_1$ is symmetric.

Definition 13.5.3 The above decomposition is called a **Symplectic-Schur Decomposition**.

The existence of the Hamiltonian-Schur decomposition and the symplectic-Schur decomposition naturally lead to the following problem: **How to obtain these decompositions in a numerically effective way by exploiting the structures of the Hamiltonian and the symplectic matrices?**

Byers (1983, 1986a) first developed such a structure-preserving method for the Hamiltonian Schur decomposition in the case the matrix Q in the Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix},$$

has rank 1. By permutation, the method is also applicable if $\text{rank}(S) = 1$.

Definition 13.5.4 *A Hamiltonian matrix H has **Hamiltonian-Hessenberg form**, if it has the zero structure of a $2n \times 2n$ upper Hessenberg matrix with the order of the last n rows and columns reversed.*

As in the standard QR iteration algorithm for the real Schur form of a matrix A , Byers' method also comes in two stages:

Stage I. The matrix H is reduced to a **Hamiltonian-Hessenberg matrix** H_H by an orthogonal-symplectic transformation.

Stage II. The Hamiltonian-Hessenberg matrix H_H is further reduced to Hamiltonian-Schur form using Hamiltonian QR iterations.

Of course, once such a reduction is done, this can immediately be used to solve the CARE. For a complete description of the method and details of numerical implementations, see Byers (1986a).

Unfortunately, in spite of several attempts, such a reduction in the general case of a Hamiltonian matrix remained a difficult problem, until the recent paper of Benner, Mehrmann and Xu (1997).

A Hamiltonian-Schur Method for the CARE ($\text{rank } S \geq 1$)

We next outline briefly the Hamiltonian-Schur method of Benner, Mehrmann and Xu (1997) for solving the CARE in the multi-input case. The method also uses symplectic-orthogonal transformations in the reduction to the Hamiltonian Schur form of the matrix H_E defined below.

The method is based on an interesting relationship between the invariant subspaces of the Hamiltonian matrix H and the extended matrix $\begin{pmatrix} 0 & H \\ H & 0 \end{pmatrix}$. It makes use of the **symplectic URV-like decomposition** that was also introduced by the authors (Benner, Mehrmann and Xu (1999)).

Theorem 13.5.3 (Symplectic-URV Decomposition)

Given a $2n \times 2n$ Hamiltonian matrix H , there exist symplectic-orthogonal matrices U_1 and U_2 such that

$$H = U_2 \begin{pmatrix} H_t & H_r \\ 0 & -H_b^T \end{pmatrix} U_1^T,$$

where H_t is an $n \times n$ upper triangular matrix and H_b is an $n \times n$ real Schur matrix.

Furthermore, the positive and negative square roots of the eigenvalues of $H_t H_b$ are the eigenvalues of H . ■

The basis of the Hamiltonian-Schur method is the following result.

Theorem 13.5.4 (Extended Hamiltonian Schur Decomposition Theorem). Suppose that the Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}$$

has no purely imaginary eigenvalues. Define

$$H_E = \begin{pmatrix} 0 & H \\ H & 0 \end{pmatrix}.$$

Then there exists, an orthogonal matrix U of order $4n$ such that

$$U^T H_E U = T = \begin{pmatrix} T_1 & T_2 \\ 0 & -T_1^T \end{pmatrix}$$

is in Hamiltonian-Schur form and no eigenvalues of T_1 have negative real parts. ■

Remark: Note that the transforming matrix U in Theorem 13.5.4 is not symplectic-orthogonal. But this non-symplectic transformation can be computed without rounding errors!

Solution of the CARE using the Extended Hamiltonian Schur Decomposition

Let the matrix U in Theorem 13.5.4 be partitioned as

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$$

where each U_{ij} is of order $2n \times 2n$. Define the matrix \hat{Y} as

$$\hat{Y} = \frac{\sqrt{2}}{2}(U_{11} - U_{21}).$$

Let Y be an orthogonal basis of $\text{Range}(\hat{Y})$. Then it has been shown (Benner, Mehrmann and Xu (1997)) that

$$\text{Range}(\hat{Y}) = \text{Inv}(H),$$

where $\text{Inv}(H)$ is the invariant subspace associated with the eigenvalues of H with negative real parts.

Furthermore, if

$$\hat{Y} = \begin{pmatrix} \hat{Y}_1 \\ \hat{Y}_2 \end{pmatrix},$$

where \hat{Y}_1 and \hat{Y}_2 are of order $n \times 2n$, then the stabilizing solution X of the CARE is given by

$$X\hat{Y}_1 = -\hat{Y}_2.$$

Note that the above equations represent an overdetermined consistent set of linear equations.

The symplectic-URV decomposition is used to compute the matrix U to achieve the Hamiltonian-Schur matrix T . Note also that it is not necessary to explicitly compute Y , if only the stabilizing solution of the CARE is sought.

The details are rather involved and we refer the readers to the paper of Benner, Mehrmann, and Xu (1997).

Efficiency and stability: The method based on the above discussion is more efficient than the Schur method. It has also been shown that the method computes the Hamiltonian-Schur form of a Hamiltonian matrix close to \tilde{H}_E , where \tilde{H}_E is permutationally similar to H_E , that is, there exists a permutation matrix P such that $PH_EP^T = \tilde{H}_E$.

Remark: Note that the above method is not structure preserving for the Hamiltonian matrix H but it is structure preserving for \tilde{H}_E .

13.5.2 The Deflating Subspace Methods

The deflating subspace methods are generalizations of the invariant subspace methods in the sense that the solutions of the Riccati equations are now computed by finding the bases for the deflating subspaces of certain matrix pencils rather than finding those of the Hamiltonian and the symplectic matrices.

For the CARE, the pencil is $P_{CARE} - \lambda N_{CARE}$, where

$$P_{CARE} = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}, \quad N_{CARE} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}. \quad (13.5.1)$$

For the DARE, the pencil is $P_{DARE} - \lambda N_{DARE}$, where

$$P_{DARE} = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}, \quad N_{DARE} = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}. \quad (13.5.2)$$

Since no inversion of A is required to form the above pencils, **this generalization is significant for the DARE**, because, as we have seen, the eigenvector and the Schur methods cannot be applied to the DARE when A is singular.

As in the case of an invariant subspace method, a basis for a deflating subspace of a pencil can be constructed either by using the generalized eigendecomposition or the generalized Schur decomposition of the pencil. As before, an eigenvector method will have numerical difficulties in case the pencil has a multiple or near-multiple eigenvalue. **We will thus skip the descriptions of the generalized eigenvector methods and describe here only the generalized Schur method for the DARE.** We leave the description of the generalized Schur method for the CARE as an exercise [Exercise 19].

The following results form a mathematical foundation for a deflating subspace method for the DARE. The results are due to Pappas, Laub and Sandell (1980).

Theorem 13.5.5 *Suppose that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Then the symplectic pencil $P_{\text{DARE}} - \lambda N_{\text{DARE}}$ does not have any eigenvalue λ with $|\lambda| = 1$.*

Proof: The proof is by contradiction.

Let $|\lambda| = 1$ be an eigenvalue of the pencil $P_{\text{DARE}} - \lambda N_{\text{DARE}}$ with the eigenvector $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \neq 0$. Then we can write

$$\begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \lambda \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}.$$

This means that

$$Az_1 = \lambda z_1 + \lambda S z_2 \quad (13.5.3)$$

$$-Qz_1 + z_2 = \lambda A^T z_2. \quad (13.5.4)$$

Premultiplying the first equation by $\bar{\lambda} z_2^*$ and postmultiplying the conjugate transpose of the second by z_1 we have

$$\bar{\lambda} z_2^* Az_1 = |\lambda|^2 z_2^* z_1 + |\lambda|^2 z_2^* S z_2 \quad (13.5.5)$$

and

$$z_2^* z_1 = z_1^* Q z_1 + \bar{\lambda} z_2^* A z_1 \quad (13.5.6)$$

Substituting (13.5.5) into (13.5.6), we obtain

$$z_2^* z_1 = z_1^* Q z_1 + |\lambda|^2 z_2^* z_1 + |\lambda|^2 z_2^* S z_2 \quad (13.5.7)$$

or

$$z_2^* S z_2 + z_1^* Q z_1 = 0 \quad (\text{since } |\lambda|^2 = 1). \quad (13.5.8)$$

Since $S = BR^{-1}B^T$, the equation (13.5.8) can be written as

$$(z_2^* B) R^{-1} (B^T z_2) + z_1^* Q z_1 = 0. \quad (13.5.9)$$

Since R is positive definite, this implies that

$$B^T z_2 = 0 \text{ and } Qz_1 = 0. \quad (13.5.10)$$

Therefore, from (13.5.3) and (13.5.4), we have $Az_1 = \lambda z_1$ and $A^T z_2 = \frac{1}{\lambda} z_2$. (Note that since $|\lambda| = 1, \lambda \neq 0$).

Thus, from (13.5.10) and from the last equation, we have $z_2^* B = 0$ and $z_2^* A = \frac{1}{\lambda} z_2^*$.

This means that for any F , $z_2^*(A - BF) = \frac{1}{\lambda} z_2^*$; that is, $\frac{1}{\lambda}$ is an eigenvalue of $A - BF$ for every F . Since (A, B) is discrete-stabilizable, this means that $z_2 = 0$. Similarly, since (A, Q) is detectable, it can be shown [Exercise 18] that $z_1 = 0$. Therefore, $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$ is a zero vector, which is a contradiction. ■

Theorem 13.5.5 together with the fact that if $\lambda \neq 0$ is an eigenvalue with multiplicity r of the pencil $P_{DARE} - \lambda N_{DARE}$, so is $\frac{1}{\lambda}$ with the same multiplicity, allows us to state the following theorem:

Theorem 13.5.6 Suppose that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Let $\lambda = 0$ be an eigenvalue of multiplicity r . Then the eigenvalues of the pencil $P_{DARE} - \lambda N_{DARE}$ can be arranged as follows (adopting the convention that the reciprocal of a zero is infinity):

$$\underbrace{0, \dots, 0}_r; \quad \underbrace{\lambda_{r+1}, \dots, \lambda_n}_{n-r}; \quad \underbrace{\frac{1}{\lambda_n}, \dots, \frac{1}{\lambda_{r+1}}}_{n-r}; \quad \underbrace{\infty, \infty, \dots, \infty}_r.$$

with $0 < |\lambda_i| < 1, i = r+1, \dots, n$. ■

MATCONTROL Note. The generalized eigenvector method for the DARE has been implemented in MATCONTROL function **ricgeigd**.

The Generalized Schur-Vector Method for the DARE

Assume that the generalized Schur form of the pencil $P_{DARE} - \lambda N_{DARE}$ has been ordered such that the generalized eigenvalues of the pencil with moduli less than 1 can be obtained from the first quarters of the matrices; that is, the orthogonal matrices Q' and Z have been computed such that

$$Q'(P_{DARE} - \lambda N_{DARE})Z = P_1 = \begin{pmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{pmatrix}$$

and

$$Q'(P_{DARE} - \lambda N_{DARE})Z = N_1 = \begin{pmatrix} N_{11} & N_{12} \\ 0 & N_{22} \end{pmatrix}$$

and the generalized eigenvalues of the pencil $P_{11} - \lambda N_{11}$ have moduli less than 1 (see below for details of how to do this).

Let $Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$. Then the columns of $\begin{pmatrix} Z_{11} \\ Z_{21} \end{pmatrix}$ form a basis for the discrete stable (deflating) subspace and the matrix $X = Z_{21}Z_{11}^{-1}$ is a unique symmetric positive semidefinite stabilizing solution of the DARE. We leave the details as an exercise [Exercise 19].

Algorithm 13.5.2 The Generalized Schur Algorithm for the DARE

Inputs: A – The $n \times n$ state matrix
 B – The $n \times m$ input matrix
 Q – The $n \times n$ state weighting
 R – The $m \times m$ control weighting matrix

Output: X - The unique $n \times n$ symmetric positive semidefinite stabilizing solution of the DARE: $A^T X A + Q - X - A^T X B (R + B^T X B)^{-1} B^T X A = 0$

Assumptions:

1. (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable.
2. $Q \geq 0, R > 0$.

Step 1. Form $P_{DARE} = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}$, $N_{DARE} = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}$.

Step 2. Transform the pencil $P_{DARE} - \lambda N_{DARE}$ to the **generalized real Schur form** using the QZ algorithm; that is find orthogonal matrices Q_1 and Z_1 such that

$$Q_1 P_{DARE} Z_1 = P_1 = \begin{pmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{pmatrix}$$

and

$$Q_1 N_{DARE} Z_1 = N_1 = \begin{pmatrix} N_{11} & N_{12} \\ 0 & N_{22} \end{pmatrix},$$

where P_1 is quasi-upper triangular and N_1 is upper triangular.

Step 3.

Reorder the above generalized real Schur form by using an orthogonal transformation, so that the pencil $P_{11} - \lambda N_{11}$ has all its eigenvalues with moduli less than 1. That is, find orthogonal matrices Q_2 and Z_2 such that $Q_2 Q_1 P_{DARE} Z_1 Z_2$ is quasi-upper triangular and $Q_2 Q_1 N_{DARE} Z_1 Z_2$ is upper triangular and moreover, the diagonal blocks corresponding to the eigenvalues with modulii less than 1 are in the upper left quarter of these matrices.

Step 4. Form $Z = Z_1 Z_2 = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$.

Step 5. Compute $X = Z_{21} Z_{11}^{-1}$; that is, solve for X : $XZ_{11} = Z_{21}$.

Example 13.5.2 Consider solving the DARE with

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = 1.$$

$$\text{Step 1. } P_{DARE} = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}, \quad N_{DARE} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 2 & 4 \end{pmatrix}.$$

Step 2. The generalized Real Schur form of the pencil $P_{DARE} - \lambda N_{DARE}$ is given by:

$Q_1(P_{DARE} - \lambda N_{DARE})Z_1 = P_1 - \lambda N_1$, where

$$P_1 = \begin{pmatrix} -5.5038 & 0.3093 & 0.7060 & 0.0488 \\ 0 & 1.4308 & 0.1222 & 0.0903 \\ 0 & 0 & 0.2665 & 0.2493 \\ 0 & 0 & 0 & 0.9530 \end{pmatrix}, \quad N_1 = \begin{pmatrix} -0.9912 & -0.3540 & 0.2965 & -0.8012 \\ 0 & -0.2842 & 0.8565 & -0.5442 \\ 0 & 0 & -1.3416 & 0.9885 \\ 0 & 0 & 0 & 5.2920 \end{pmatrix}$$

Step 3. The eigenvalues with modulii less than 1 are :

$$\frac{P_1(3,3)}{N_1(3,3)} = -0.1986 \text{ and } \frac{P_1(4,4)}{N_1(4,4)} = 0.1801.$$

Step 4. The matrix $\begin{pmatrix} Z_{11} \\ Z_{21} \end{pmatrix}$ is given by

$$\begin{pmatrix} Z_{11} \\ Z_{21} \end{pmatrix} = \begin{pmatrix} 0.5518 & -0.1074 \\ -0.3942 & 0.0847 \\ 0.6400 & 0.4499 \\ -0.3614 & 0.8825 \end{pmatrix}$$

Step 5. $X = Z_{21} Z_{11}^{-1} = \begin{pmatrix} 54.9092 & 75.2247 \\ 75.2247 & 106.1970 \end{pmatrix}$ is the stabilizing solution.

Implementational Details

The reduction to the generalized real Schur form can be achieved using the QZ algorithm, as described in Chapter 4.

Unfortunately, however, the eigenvalues might appear in any arbitrary order. Some reordering needs to be done. A systematic way to do this is as follows:

First, check if the last eigenvalue in the upper left quarter has modulus less than 1, if not, move it to the last position in the lower right quarter. Check the next eigenvalue now in the upper left quarter, if it does not have modulus less than 1, move it to the next position in the lower right quarter.

Note that each move is equivalent to finding a pair of orthogonal matrices such that pre- and post-multiplications by these matrices perform the necessary change.

The process can be continued until all the n eigenvalues with moduli greater than 1 have been moved to the lower right quarter and the upper left quarter contains only the eigenvalues with moduli less than 1.

There is also a slightly more efficient algorithm (Sima (1996), pp. 262-264) for ordering the eigenvalues of the pencil $P_{DARE} - \lambda N_{DARE}$.

There exists FORTRAN Routines, developed by Van Dooren (1982) to compute deflating subspaces with specified spectrum. These subroutines are available as Algorithm 590–DSUBSP and EXCHQZ in ACM software library. Also, the LAPACK package (Anderson et al. (1995)) includes the routine STGSEN, which performs a specified reordering of the eigenvalues of the generalized real Schur form.

Numerical stability and scaling. It can be shown (see Petkov et al. (1989)) that the generalized Schur method may yield inaccurate results if the DARE is not properly scaled. For a scaling strategy that can be used to overcome this problem, see Gudmundsson et al. (1992) and Benner (1997).

The Generalized Schur Methods Without Explicit Computation of the Inverse of the Control Weighting Matrix R .

All the methods we have considered so far require the explicit computation of the inverse of the control weighting matrix R . **These methods, therefore, may not yield accurate solutions when R is severely ill-conditioned.**

For example, consider the following example from Arnold and Laub (1984):

$$A = \begin{pmatrix} -0.1 & 0 \\ 0 & -0.02 \end{pmatrix}, B = \begin{pmatrix} 0.1 & 0 \\ 0.001 & 0.01 \end{pmatrix}, Q = \begin{pmatrix} 100 & 1000 \\ 1000 & 10000 \end{pmatrix}, R = \begin{pmatrix} 1 + \epsilon & 1 \\ 1 & 1 \end{pmatrix}.$$

The pair (A, B) is controllable. The matrix R becomes progressively ill-conditioned as $\epsilon \rightarrow 0$. The CARE with the above data was solved by Arnold and Laub, using RICPACK, a software package especially designed for solving Riccati equations. It was shown that the accuracy of the solution deteriorated as R became more and more ill-conditioned. For $\epsilon = 10^{-16}$ the relative accuracy was of 10^{-1} only.

In this case, an **inverse-free generalized Schur method**, that avoids computations of R^{-1} is useful.

The Continuous-Time Case

First, we observe that the Hamiltonian eigenvalue problem $Hx = \lambda x$ associated with the CARE, can be replaced by the eigenvalue problem for the extended $(2n + m) \times (2n + m)$ pencil:

$$P_{CARE}^E - \lambda N_{CARE}^E,$$

where $P_{CARE}^E = \begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix}$, and $N_{CARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix}$.

(Note that this pencil does not involve R^{-1}). The solution of the CARE can now be obtained by constructing a basis of the stable deflating subspace of this pencil. It was further observed by Van Dooren (1981) that this $(2n + m) \times (2n + m)$ can be compressed, using an orthogonal factorization of the matrix $\begin{pmatrix} R \\ B \end{pmatrix}$, into a $2n \times 2n$ pencil, without affecting the deflating subspaces. Thus, if

$$\begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} \begin{pmatrix} R \\ B \end{pmatrix} = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix},$$

then instead of considering the $(2n + m) \times (2n + m)$ pencil $P_{CARE}^E - \lambda N_{CARE}^E$, we consider the $2n \times 2n$ compressed pencil $P_{CARE}^{EC} - \lambda N_{CARE}^{EC}$, where $P_{CARE}^{EC} = \begin{pmatrix} W_{22}A & W_{21}B^T \\ -Q & -A^T \end{pmatrix}$ and $N_{CARE}^{EC} = \begin{pmatrix} W_{22} & 0 \\ 0 & I \end{pmatrix}$.

This leads to the following algorithm:

Algorithm 13.5.3 Inverse-Free Generalized Schur Algorithm for the CARE.

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix ($m \leq n$)

Q - The $n \times n$ state weighting matrix

R - The $m \times m$ control weighting matrix

Output: X - The unique symmetric positive semidefinite stabilizing solution of the CARE

Assumptions: 1) (A, B) is stabilizable, and (A, Q) is detectable.

2) $Q \geq 0, R > 0$.

Step 1. Find the QR factorization of the matrix $\begin{pmatrix} R \\ B \end{pmatrix}$:

$$W \begin{pmatrix} R \\ B \end{pmatrix} = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}.$$

Partition $W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$, where W_{22} is an $n \times n$ matrix.

Step 2. Form P_{CARE}^{EC} and N_{CARE}^{EC} as shown above.

Step 3. Find the ordered generalized Schur form of the pencil $P_{CARE}^{EC} - \lambda N_{CARE}^{EC}$ using the QZ algorithm; that is, find orthogonal matrices Q_1 and Z such that $Q_1(P_{CARE}^{EC} - \lambda N_{CARE}^{EC})Z = \tilde{M} - \lambda \tilde{N}$; where \tilde{M} and \tilde{N} are, respectively, quasi-upper and upper triangular matrices, and the n eigenvalues with negative real parts appear first.

Step 4. Compute $X = Z_{21}Z_{11}^{-1}$ where $Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$. ■

Remark: In his paper, Van Dooren (1981) described the compression technique by using an orthogonal factorization of the matrix $\begin{pmatrix} B \\ 0 \\ R \end{pmatrix}$.

Instead, here we have used (an equivalent) factorization of $\begin{pmatrix} R \\ B \end{pmatrix}$ in the form $\begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}$, so that a standard QR factorization algorithm can be used to achieve this factorization.

Example 13.5.3

$$A = \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = 10^{-10}$$

Step 1.

$$W = \left(\begin{array}{c|cc} -0.0000 & -1.0000 & 0 \\ \hline -1.0000 & 0.0000 & 0 \\ 0 & 0 & 1.0000 \end{array} \right) = \left(\begin{array}{c|c} W_{11} & W_{12} \\ \hline W_{21} & W_{22} \end{array} \right)$$

Step 2.

$$P_{CARE}^{EC} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & -2 & -1 \\ 0 & -1 & 1 & 0 \end{pmatrix}$$

$$N_{CARE}^{EC} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Step 3.

$$Z = \begin{pmatrix} -1.0000 - 0.0028i & 0.0000 + 0.0000i & -0.0000 + 0.0000i & -0.0000 + 0.0000i \\ 0.0000 + 0.0000i & 0.7071 + 0.0025i & -0.7071 + 0.0044i & 0.0000 - 0.0000i \\ -0.0000 - 0.0000i & 0.0000 + 0.0000i & 0.0000 - 0.0000i & 1.0000 - 0.0000i \\ 0.0000 + 0.0000i & 0.7071 + 0.0025i & 0.7071 - 0.0044i & -0.0000 + 0.0000i \end{pmatrix}$$

Step 4.

$$X \text{ (in Long Format)} = \begin{pmatrix} 0.00001000030018 & 0.00000999990018 \\ 0.00000999990018 & 1.00001000029721 \end{pmatrix}.$$

Verify: The residual norm= 7.357×10^{-8}

The Discrete-Time Case

The discrete problem is analogous. Here we consider the $(2n+m) \times (2n+m)$ pencil $P_{DARE}^E - \lambda N_{DARE}^E$, where

$$P_{DARE}^E = \begin{pmatrix} A & 0 & -B \\ -Q & -I & 0 \\ 0 & 0 & R \end{pmatrix},$$

and

$$N_{DARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & A^T & 0 \\ 0 & B^T & 0 \end{pmatrix}.$$

This pencil is then compressed into the $2n \times 2n$ pencil $P_{DARE}^{EC} - \lambda N_{DARE}^{EC}$ where

$$P_{DARE}^{EC} = \begin{pmatrix} W_{22}A & 0 \\ -Q & -I \end{pmatrix},$$

and

$$N_{DARE}^{EC} = \begin{pmatrix} W_{22} & W_{21}B^T \\ 0 & A^T \end{pmatrix},$$

by taking the QR factorization of the matrix $\begin{pmatrix} R \\ -B \end{pmatrix}$:

$$W \begin{pmatrix} R \\ -B \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \text{ where } W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}.$$

This leads to the following algorithm:

Algorithm 13.5.4 Inverse-free Generalized Schur Method for the DARE.

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix ($m \leq n$)

Q - The $n \times n$ state weighting matrix

R - The $m \times m$ control weighting matrix

Output: X - The unique symmetric positive semidefinite stabilizing solution of the DARE.

Assumptions: 1) (A, B) is stabilizable and (A, Q) is detectable.

2) $Q \geq 0, R > 0$.

Step 1. Find the QR factorization of $\begin{pmatrix} R \\ -B \end{pmatrix}$; that is find an orthogonal matrix W such that

$$W \begin{pmatrix} R \\ -B \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}.$$

$$\text{Partition } W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}.$$

$$\text{Step 2. Form } P_{DARE}^{EC} = \begin{pmatrix} W_{22}A & 0 \\ -Q & -I \end{pmatrix}, \quad N_{DARE}^{EC} = \begin{pmatrix} W_{22} & W_{21}B^T \\ 0 & A^T \end{pmatrix}.$$

Step 3. Compute the **ordered generalized Schur form** of the pencil $P_{DARE}^{EC} - \lambda N_{DARE}^{EC}$, using the QZ algorithm followed by some ordering procedure so that the eigenvalues of moduli less than 1 appear in the first quarter; that is, find orthogonal matrices Q_1 and Z such that $Q_1(P_{DARE}^{EC} - \lambda N_{DARE}^{EC})Z = \tilde{P} - \lambda \tilde{N}$ and the n eigenvalues with modulii less than 1 appear first.

$$\text{Step 4. Form } X = Z_{21}Z_{11}^{-1} \text{ where } Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix},$$

Example 13.5.4 Consider solving the DARE with

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}, R = 1.$$

Step 1.

$$W = \left(\begin{array}{c|cc} -0.7071 & 0 & 0.7071 \\ \hline 0 & 1.0000 & 0 \\ 0.7071 & 0 & 0.7071 \end{array} \right) = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$$

Step 2.

$$P_{DARE}^{EC} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -2 & -1 & 0 \\ -2 & -4 & 0 & -1 \end{pmatrix}$$

$$N_{DARE}^{EC} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.7071 & 0 & 0.7071 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Step 3.

$$Z = \begin{pmatrix} 0.8615 & -0.2781 & 0.3731 & -0.2034 \\ -0.3290 & 0.3256 & 0.8231 & -0.3290 \\ 0.2034 & 0.3731 & 0.2781 & 0.8615 \\ 0.3290 & 0.8231 & -0.3256 & 0.9329 \end{pmatrix}$$

Step 4.

$$X = \begin{pmatrix} 1.0000 & 2.0000 \\ 2.0000 & 4.2361 \end{pmatrix}$$

Verify: The residual norm = 7.772×10^{-16} .

MATLAB Note. MATLAB functions **care** and **dare** solve the CARE and DARE, respectively, using generalized Schur methods.

13.5.3 The Matrix Sign Function Methods

Let A be an $n \times n$ matrix with **no zero or purely imaginary eigenvalues**.

Let

$$J = X^{-1}AX = D + N,$$

where $D = \text{diag}(d_1, \dots, d_n)$ and N is nilpotent and commutes with D , be the Jordan Canonical Form of A . Then the matrix sign-function of A is defined as:

$\text{Sign}(A) = X \text{diag}(\text{sign}(d_1), \text{sign}(d_2), \dots, \text{sign}(d_n)) X^{-1}$, where

$$\text{sign}(d_i) = \begin{cases} 1 & \text{if } \text{Re}(d_i) > 0 \\ -1 & \text{if } \text{Re}(d_i) < 0 \end{cases}$$

Some important properties of $\text{Sign}(A)$ are [Exercise 17]:

1. $\text{Sign}(A)$ has the same stable invariant subspace as A .
2. The eigenvalues of $\text{Sign}(A)$ are ± 1 , depending upon the sign of the corresponding eigenvalues of A .
3. The range of $\text{Sign}(A) - I$ is the stable invariant subspace of A .
4. The eigenvectors of $\text{Sign}(A)$ are the eigenvectors and principal vectors of A .
5. $\text{Sign}(TAT^{-1}) = T\text{Sign}(A)T^{-1}$.

In this section, we will describe algorithms for solving the CARE and the DARE using matrix sign functions. Before doing so, let's first describe an algorithm for computing $\text{Sign}(A)$.

The basic sign-function algorithm is:

$$\begin{aligned} Z_0 &= A \\ Z_{k+1} &= \frac{1}{2} (Z_k + Z_k^{-1}), \quad k = 0, 1, \dots \end{aligned}$$

It can be shown that the sequence $\{Z_k\}$ converges to $\text{Sign}(A)$ quadratically.

The initial convergence can, however, be very slow. Byers (1987) has shown that the convergence can be accelerated if Z_k is scaled by $|\det(Z_k)|^{1/n}$. For a discussion of scaling see Kenney and Laub (1992).

Thus a **practical algorithm** for computing $\text{Sign}(A)$ is:

Algorithm 13.5.5 Computing $\text{Sign}(A)$.

Input: An $n \times n$ matrix A

Output: $\text{Sign}(A)$, the matrix sign-function of A .

Step 1. Set $Z_0 = A$

Step 2. For $k = 0, 1, 2, \dots$, do until convergence

Compute $c = |\det Z_k|^{1/n}$.

Compute $Z_{k+1} = \frac{1}{2c} (Z_k + c^2 Z_k^{-1})$

End

Stopping criteria. The algorithm can be terminated if

- (i) the norm of the difference between two successive iterates is small enough
or
- (ii) the number of iterations exceeds the maximum number prescribed.

The Matrix Sign-Function Method for the CARE

The mathematical basis for the matrix sign-function method for the CARE is the following theorem.

Theorem 13.5.7 (Roberts (1971)). Let H be the Hamiltonian matrix (13.2.1) associated with the CARE: $XA + A^T X + Q - XSX = 0$.

Let (A, B) be stabilizable and let (A, Q) be detectable.

Let $\text{Sign}(H) = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$, where W_{ij} are $n \times n$ real matrices.

Then the unique symmetric positive semidefinite stabilizing solution X of the CARE is given by the following **overdetermined** consistent linear systems:

$$\begin{pmatrix} W_{12} \\ W_{22} + I \end{pmatrix} X = - \begin{pmatrix} W_{11} + I \\ W_{21} \end{pmatrix}.$$

Proof:

Define

$$T = \begin{pmatrix} I & Y \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ -X & I \end{pmatrix} = \begin{pmatrix} I - YX & Y \\ -X & I \end{pmatrix},$$

where Y satisfies

$$(A - SX)Y + Y(A - SX)^T = -S.$$

An easy computation then shows that

$$THT^{-1} = \begin{pmatrix} A - SX & 0 \\ 0 & -(A - SX)^T \end{pmatrix}.$$

Note that $T^{-1} = \begin{pmatrix} I & -Y \\ X & I - XY \end{pmatrix}$.

Then, using Property 5 of the Sign-Function matrix we obtain

$$\begin{aligned} \text{Sign}(H) &= T^{-1} \text{Sign} \begin{pmatrix} A - SX & 0 \\ 0 & -(A - SX)^T \end{pmatrix} T \\ &= T^{-1} \begin{pmatrix} -I & 0 \\ 0 & I \end{pmatrix} T \quad (\text{since } A - SX \text{ is asymptotically stable}) \\ &= \begin{pmatrix} 2YX - I & -2Y \\ 2XYX - 2X & I - 2XY \end{pmatrix}. \end{aligned}$$

Thus $\text{Sign}(H) + I_{2n} = \begin{pmatrix} 2YX & -2Y \\ 2XYX - 2X & 2I - 2XY \end{pmatrix}$

or $\begin{pmatrix} W_{11} + I & W_{12} \\ W_{21} & W_{22} + I \end{pmatrix} = \left(\begin{bmatrix} 2Y \\ 2(XY - I) \end{bmatrix} X, -\begin{bmatrix} 2Y \\ 2(XY - I) \end{bmatrix} \right)$.

Comparing now both sides of the equation, we see that X must satisfy:

$$\begin{pmatrix} W_{12} \\ W_{22} + I \end{pmatrix} X = -\begin{pmatrix} W_{11} + I \\ W_{21} \end{pmatrix}.$$

■

Symmetric Version of the Matrix Sign-Function Algorithm

Theorem 13.5.7 yields a computational method to solve the CARE. However, the convergence can be painfully slow. The method can be made more efficient by using the following trick (Bierman (1984), Byers (1987)) in which one works only with symmetric matrices.

Define

$$W_0 = JH = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} = \begin{pmatrix} -Q & -A^T \\ -A & S \end{pmatrix}.$$

The matrix W_0 is symmetric.

Now compute $Sign(H)$ by performing the following iterations:

$$W_{k+1} = \frac{1}{2c_k} (W_k + c_k^2 JW_k^{-1} J), k = 0, 1, 2, \dots$$

Then each W_k is symmetric and $\lim_{k \rightarrow \infty} W_k = Jsign(H)$.

The parameter c_k is chosen to enhance the rate of convergence, as before.

Let $JSign(H) = Y = \begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix}$. Then

$$Sign(H) = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} = J^T Y,$$

The equations

$$\begin{pmatrix} W_{12} \\ W_{22} + I \end{pmatrix} X = -\begin{pmatrix} W_{11} + I \\ W_{21} \end{pmatrix}$$

then become

$$\begin{pmatrix} Y_{22} \\ Y_{12} + I \end{pmatrix} X = \begin{pmatrix} I - Y_{21} \\ -Y_{11} \end{pmatrix}.$$

This leads to the following symmetric version of the matrix sign-function algorithm for the CARE:

Algorithm 13.5.6 The Matrix Sign-Function Algorithm for the CARE.

Inputs: A - The $n \times n$ state matrix
 B - The $n \times m$ input matrix
 Q - The $n \times n$ state weighting matrix
 R - The $m \times m$ control weighting matrix
 ϵ - Error tolerance

Output: X - The symmetric positive semidefinite stabilizing solution of the CARE:

$$A^T X + X A - X B R^{-1} B^T X + Q = 0$$

Assumptions: 1) (A, B) is stabilizable and (A, Q) is detectable.
2) $Q \geq 0, R > 0$.

Step 1. 1.1 Form $S = BR^{-1}B^T$

1.2 Define $J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$. Form $W = JH = \begin{pmatrix} -Q & -A^T \\ -A & S \end{pmatrix}$

Step 2. For $k = 1, 2, \dots$ do until convergence with the given tolerance ϵ

$$c = |\det W|^{1/2n}$$

$$W = \frac{1}{2c}(W + c^2 JW^{-1}J)$$

Step 3. Partition $W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$, where each W_{ij} is of order n .

Step 4. Form $M = \begin{pmatrix} W_{22} \\ W_{12} + I_n \end{pmatrix}$, $N = \begin{pmatrix} I - W_{21} \\ -W_{11} \end{pmatrix}$.

Step 5. Solve for $X : MX = N$.

Example 13.5.5 Consider solving the CARE using Algorithm 13.5.6 with $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$,

$$B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, R = 1.$$

Step 1. $S = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$, $H = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 \end{pmatrix}$, $W_0 = JH = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Step 2. $W_1 = \frac{1}{2}(W_0 + JW_0^{-1}J) = \begin{pmatrix} -1 & 0 & 0 & -0.5 \\ 0 & -1 & -0.5 & 0 \\ 0 & -0.5 & 0.5 & 0 \\ -0.5 & 0 & 0 & 0.5 \end{pmatrix}$,

$$c = |\det(W_1)|^{\frac{1}{4}} = 0.8660$$

$$W_2 = \frac{1}{2c}(W_1 + c^2 JW_1^{-1}J) = \begin{pmatrix} -1.1547 & 0 & 0 & -0.5774 \\ 0 & -1.1548 & -0.5774 & 0 \\ 0 & -0.5774 & 0.5774 & 0 \\ -0.5774 & 0 & 0 & 0.5774 \end{pmatrix}.$$

(Note that each $W_i, i = 0, 1, 2$ is symmetric)

Step 3. $J \operatorname{Sign}(H) = W_2 = W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$.

Step 5. $X = \begin{pmatrix} 1.7321 & 1 \\ 1 & 1.7321 \end{pmatrix}$.

Verify: The residual norm $= 9.9301 \times 10^{-16}$

Example 13.5.6 Consider now solving the CARE using Algorithm 13.5.6 with the following data:

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, R = 1.$$

Step 1.

$$W_0 = \begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 & 3 \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 1 & 1 & 1 \\ 0 & 0 & 3 & 1 & 1 & 1 \end{pmatrix}$$

Step 2. After five iterations, $\| W_5 - W_4 \| / \| W_4 \| = 6.4200 \times 10^{-15}$ (The readers are asked to verify this by carrying out 5 iterations).

Step 3.

$$W \equiv W_5$$

Step 5.

$$X = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Verify: The residual norm $= 3.1602 \times 10^{-16}$.

Flop-count and Stability. It can be shown that Algorithm 13.5.6 requires about $4n^3$ flops per iteration. **The algorithm is not stable in general (Byers (1986b)),** unless used with an iterative refinement technique such as Newton's method (see **Section 13.5.4**).

MATCONTROL Note. Algorithm 13.5.6 has been implemented in MATCONTROL function **ricsgnc**.

The Matrix Sign-Function Method for the DARE.

The matrix sign-function method for the CARE described in the previous section can now be applied to solve the DARE by converting the symplectic matrix M to the Hamiltonian matrix H using the bilinear transformation:

$$H = (M + I)^{-1} (M - I).$$

Because A needs to be nonsingular, the method is not applicable if A is singular, and is not numerically effective when A is ill-conditioned.

Avoiding Explicit Inversion of A

The explicit inversion of A , however, may be avoided, by using the following simple trick (Gardiner and Laub (1986)).

Write

$$M = N^{-1}P,$$

$$\text{where } N = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}, \text{ and } P = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}.$$

Then it can be shown that even if A is singular, the matrix $(P + N)$ is invertible and the matrix H can be expressed as $H = (P + N)^{-1}(P - N)$.

Algorithm 13.5.7 The Matrix Sign-Function Algorithm for the DARE.

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix

Q - The $n \times n$ state weighting matrix

R - The $m \times m$ control weighting matrix

Output: X - The unique symmetric positive semidefinite stabilizing solution X of the DARE:

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$

Assumptions: 1) (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable,
2) $Q \geq 0, R > 0$.

Step 1. Form $S = BR^{-1}B^T$, $N = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}$, $P = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}$.

Step 2. Form $H = (P + N)^{-1}(P - N)$.

Step 3. Apply the matrix sign-function algorithm for the continuous-time algebraic Riccati equation (Algorithm 13.5.6) with H in Step 2.

Example 13.5.7 Consider solving the DARE using Algorithm 13.5.7 with $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$,

$$B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, R = 1.$$

Step 1. $S = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$, $N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, $P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$.

Step 2. $H = \begin{pmatrix} -0.3333 & 0.6667 & -0.6667 & 0.6667 \\ -0.6667 & 0.3333 & 0.6667 & -0.6667 \\ -1.3333 & 0.6667 & 0.3333 & 0.6667 \\ 0.6667 & -1.3333 & -0.6667 & -0.3333 \end{pmatrix}$

Step 3. $X = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$.

Verify: The residual norm = 6.7195×10^{-16} .

MATCONTROL Note. Algorithm 13.5.7 has been implemented in MATCONTROL function **ricsgnd**.

13.5.4 Newton's Methods

Recall that the classical Newton's method for finding a root x of $f(x) = 0$ can be stated as follows:

- (i) Choose x_0 , an initial approximation to x .
- (ii) Generate a sequence of approximations $\{x_i\}$ defined by

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad i = 0, 1, 2, \dots \quad (13.5.11)$$

Then, whenever x_0 is chosen close enough to x , the sequence $\{x_i\}$ converges to the root x and the convergence is quadratic if $f'(x) \neq 0$. Newton's methods for the CARE and DARE can similarly be developed.

Newton's Method for the Continuous-time Algebraic Riccati Equation

Consider first the CARE: $XA + A^T X - XBR^{-1}B^T X + Q = 0$.

Starting from an initial approximate solution X_0 , the computed solutions are iteratively refined until convergence occurs; and this is done by solving a Lyapunov equation at each iteration. The way how the Lyapunov equations arise can be explained as follows. Write $X = X_0 + (X - X_0)$. Substituting this into the CARE we have

$$\begin{aligned} & (A - BR^{-1}B^T X_0)^T X + X(A - BR^{-1}B^T X_0) \\ &= -X_0 BR^{-1}B^T X_0 - Q + (X - X_0)BR^{-1}B^T(X - X_0) \end{aligned}$$

Assuming that $X - X_0$ is small (that is, the initial approximate solution is good), we can neglect the last term on the right hand side of the above equation. Thus we obtain the following Lyapunov equation for the next approximation X_1 :

$$(A - BR^{-1}B^T X_0)^T X_1 + X_1(A - BR^{-1}B^T X_0) = -X_0 BR^{-1}B^T X_0 - Q$$

Assuming that X_1 is a better approximation than X_0 (that is $\|X - X_1\| < \|X - X_0\|$), the process can be continued until the convergence occurs, if there is convergence.

The above discussion immediately suggests the following **Newton method for the CARE**: (Kleinman (1968)):

Step 1. Choose an initial approximation X_0 .

Step 2. Compute $\{X_k\}$ iteratively by solving the Lyapunov equation:

$$(A - SX_k)^T X_{k+1} + X_{k+1}(A - SX_k) = -X_k SX_k - Q, \quad k = 0, 1, 2, \dots,$$

where $S = BR^{-1}B^T$.

Step 3. Continue until and if convergence occurs.

Newton's method, as stated above, is not in the familiar form. However, the above steps can be easily reorganized to obtain Newton's method in the familiar form (see Benner (1997), Hammarling (1982) and Lancaster and Rodman (1995) for details).

To do this, let's define

$$R_C(X) = XA + A^T X - XSX + Q,$$

where $S = BR^{-1}B^T$.

Now, the Fréchet derivative of $R_C(X)$ is given by

$$R'_X(Z) := (A - SX)^T Z + Z(A - SX)$$

Thus, Newton's method for $R_C(X) = 0$ is

$$R'_{X_i}(\Delta_i) + R_C(X_i) = 0, i = 0, 1, 2, \dots$$

$$X_{i+1} = X_i + \Delta_i$$

The above observation leads to the following Newton algorithm for the CARE.

Algorithm 13.5.8 Newton's Method for the CARE

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix

Q - The $n \times n$ state weighting matrix

R - The $m \times m$ control weighting matrix

Output: The set $\{X_k\}$ converging to an approximate stabilizing solution matrix X of the CARE.

Assumptions: 1) (A, B) is stabilizable and (A, Q) is detectable.

2) $Q \geq 0$, $R > 0$.

Step 1. Set $S = BR^{-1}B^T$.

Step 2. Choose an initial approximate solution $X_0 = X_0^T$ such that $A - SX_0$ is stable.

Step 3. Construct the sequence of solutions $\{X_i\}$ as follows:

For $i = 0, 1, 2, \dots$ do until convergence occurs

3.1 Compute $A_i = A - SX_i$

3.2 Compute $R_C(X_i) = A^T X_i + X_i A + Q - X_i S X_i$

3.3 Solve the Lyapunov equation for Δ_i : $A_i^T \Delta_i + \Delta_i A_i + R_C(X_i) = 0$.

3.4 Compute $X_{i+1} = X_i + \Delta_i$

End

Remarks: The above form of Newton's method is usually known as **Newton's Method in incremental form**. This form has some computational advantages over that presented in the beginning of this section in the sense that, in general, more accurate answers can be expected. This is because, in the incremental form algorithm, we solve the Lyapunov equation for the increment Δ_i and not for the solution directly and therefore, the solution X_i will have more correct digits.

The proof of the following theorem can be found in Lancaster and Rodman (1995, page 232-233). It gives conditions under which the above iterates converge.

Theorem 13.5.8 (Convergence of Newton's Method for the CARE).

Let (A, B) be stabilizable and (A, Q) be detectable. Let X_0 be an approximate stabilizing solution and let X be a unique symmetric positive semidefinite stabilizing solution X of the CARE. Then the matrices A_i and $X_i, i = 0, 1, \dots$, constructed by the above algorithm are such that

- (i) All A_i are stable; that is, all iterates X_i are stabilizing.
- (ii) $0 \leq X \leq \dots \leq X_{i+1} \leq X_i \leq \dots \leq X_1$.
- (iii) $\lim_{i \rightarrow \infty} X_i = X$, where X is the unique symmetric positive-semidefinite stabilizing solution of the CARE.
- (iv) There exists a constant $c > 0$ such that $\|X_{i+1} - X\| \leq c \|X_i - X\|^2$, for $i \geq 1$; that is, the sequence $\{X_i\}$ converges quadratically.

■

Remark: Note that (ii) and (iv) are also true for $i = 0$, if X_0 is close enough to X . But, in general, they are not true for $i = 0$ (see the counter-example later).

Stopping Criterion

The following can be used as a stopping criterion.

Stop the iteration if

I. For a certain value of k and the prescribed *tolerance* ϵ

$$\frac{\|X_{k+1} - X_k\|_F}{\|X_k\|_F} \leq \epsilon,$$

Or

II. The number of iterations k exceeds a prescribed number N .

If a condition-number estimator for the CARE is available, then criterion I can be replaced by the following more appropriate stopping criterion: Stop the iteration if

$$\frac{\|X_{k+1} - X_k\|_F}{\|X_k\|_F} \leq \mu \kappa_{CARE}^E,$$

where κ_{CARE}^E denotes an estimate of the κ_{CARE} and μ is the machine precision.

Example 13.5.8 Consider solving the CARE using Newton's method (**Algorithm 13.5.8**) with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad R = 1.$$

Step 1. $S = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

Step 2. $X_0 = \begin{pmatrix} 0.4 & 0.1 & 0.1 \\ 0.1 & 0.3 & 0.0 \\ 0.1 & 0 & 0.2 \end{pmatrix}$

Step 3.

$$i = 0$$

$$\Delta_0 = \begin{pmatrix} -0.0248 & -0.0302 & -0.0369 \\ -0.0302 & -0.0426 & 0.0103 \\ -0.0369 & 0.0103 & -0.0224 \end{pmatrix}$$

$$X_1 = X_0 + \Delta_0 = \begin{pmatrix} 0.3752 & 0.0698 & 0.0631 \\ 0.0698 & 0.2574 & 0.0103 \\ 0.0631 & 0.0103 & 0.1776 \end{pmatrix}$$

Relative Change: $\frac{\|X_1 - X_0\|}{\|X_0\|} = 0.1465$

$$i = 1.$$

$$\Delta_1 = \begin{pmatrix} -0.0020 & -0.0015 & -0.0010 \\ -0.0015 & -0.0011 & -0.0008 \\ -0.0010 & -0.0008 & -0.0005 \end{pmatrix}$$

$$X_2 = X_1 + \Delta_1 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\frac{\|X_2 - X_1\|}{\|X_1\|} = 0.0086.$

$$i = 2$$

$$\Delta_2 = 10^{-5} \begin{pmatrix} -0.4561 & -0.3864 & -0.2402 \\ -0.3864 & -0.3311 & -0.2034 \\ -0.2402 & -0.2034 & -0.1265 \end{pmatrix}$$

$$X_3 = X_2 + \Delta_2 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\frac{\|X_3 - X_2\|}{\|X_2\|} = 2.1709 \times 10^{-5}.$

MATHCONTROL Note. Algorithm 13.5.8 has been implemented in MATCONTROL function **ricnwtnc**.

Convergence: We know that there exist infinitely many X_0 for which $A - SX_0$ is stable. *The choice of proper X_0 is crucial.* If the initial solution matrix X_0 is not close enough to the true solution X , then, as in the case of scalar Newton's method, the convergence can be painfully slow. *The method even might converge to a nonstabilizing solution in the presence of round-off errors.* Things might go wrong even at the first step. To see this, let's consider the following example from Kenney, Laub and Wette (1990):

$$A = 0, B = Q = I, R = I.$$

The exact solution is $X = I$. Let $X_0 = \epsilon I$, where $\epsilon > 0$ is a small positive number. Then

$$A - BB^T X_0 = -\epsilon I$$

is stable for all $\epsilon > 0$ and the initial error is $\|X - X_0\| = 1 - \epsilon \cong 1$ for small ϵ . However,

$$X_1 = \frac{1 + \epsilon^2}{2\epsilon} I,$$

and

$$\|X - X_1\| \simeq \frac{1}{2\epsilon},$$

which is quite large. Thus, even though the errors at subsequent steps decreases, a large number of steps will be needed for the error made at the first step to die out.

Some conditions guaranteeing convergence from the first step on have been given by Kenney, Laub, and Wette (1990). This is stated in the following Theorem (**assuming that $R = I_{m \times m}$**).

Theorem 13.5.9 *Let X_0 be an initial approximation such that $A - BB^T X_0$ is stable and assume that $\|X - X_0\| < \frac{1}{3\|B\|^2\|\Omega^{-1}\|}$, where $\Omega(Z) = (A - BB^T X)^T Z + Z(A - BB^T X)$, then $\|X - X_1\| \leq \|X - X_0\|$, with equality only when $X_0 = X$.*

Flop-Count. Newton's method is iterative; therefore, an exact flop count cannot be given. However, if the Schur method is used to solve the Lyapunov equations at each iteration, then about $40n^3$ flops are needed per iteration.

Stability. Since the principal computational task in Newton's Method is the solution of a Lyapunov matrix equation at each iteration, **the method can be shown to be stable if a numerically stable method such as the Schur method is used to solve the Lyapunov equation.** Specifically, if \hat{X} is the computed solution obtained by Newton's method, then it can be shown (Petkov et al. (1991)) that

$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} \leq \mu\kappa_{CARE},$$

where κ_{care} is the condition number of the CARE. **That is, the method does not introduce more errors than what is already inherent in the problem.**

Modified Newton's Methods

Several modifications of Newton's methods for the algebraic Riccati equations have been obtained in recent years (Benner and Byers (1994, 1998), Guo and Lancaster (1998), Guo (1998), etc.). We just state in the following the line search modification of Newton's method by Benner and Byers (1998).

Newton's Method With Line Search

The performance of Newton's method can be improved by using an optimization technique, called **line search**.

The idea is to take a Newton step at each iteration in the direction so that $\|R_C(X_{i+1})\|_F^2$ is minimized. Thus the iteration

$$X_{i+1} = X_i + \Delta_i$$

in Step 3 of Newton's method will be replaced by

$$X_{i+1} = X_i + t_i \Delta_i,$$

where t_i is a real scalar to be chosen so that $\|R_C(X_i + t_i \Delta_i)\|_F^2$ will be minimized.

This is equivalent to minimizing

$$\begin{aligned} f_i(t) &= \text{Trace}(R_C(X_i + t\Delta_i)^T R_C(X_i + t\Delta_i)) = \text{Trace}(R_C(X_i + t\Delta_i)^2) \\ &= \alpha_i(1-t)^2 - 2\beta_i(1-t)t^2 + \nu_i t^4 \end{aligned}$$

where

$$\begin{aligned} \alpha_i &= \text{Trace}(R_C(X_i)^2) \\ \beta_i &= \text{Trace}(R_C(X_i)V_i) \\ \nu_i &= \text{Trace}(V_i^2), \\ V_i &= \Delta_i S \Delta_i. \end{aligned}$$

It can be shown [see Benner (1997), Benner and Byers (1998)] that the function $f_i(t)$ has a local minimum at some value $t_i \in [0, 2]$.

We thus have the following modified Newton's algorithm.

Algorithm 13.5.9 *Newton's Method With Line Search for the CARE*

Inputs: Same as in Algorithm 13.5.8

Output: Same as in Algorithm 13.5.8

Assumptions: Same as in Algorithm 13.5.8.

Step 1. Same as in Algorithm 13.5.8.

Step 2. Same as in Algorithm 13.5.8.

Step 3. For $i = 0, 1, 2, \dots$ do until convergence occurs

3.1 Same as in Algorithm 13.5.8.

3.2 Same as in Algorithm 13.5.8.

3.3 Same as in Algorithm 13.5.8.

3.4 Compute $V_i = \Delta_i S \Delta_i$

3.5 Compute α_i, β_i , and ν_i of f_i as given above.

3.6 Compute $t_i \in [0, 2]$ such that $f_i(t_i) = \min_{t \in [0, 2]} f_i(t)$.

3.7 Compute $X_{i+1} = X_i + t_i \Delta_i$.

End.

Example 13.5.9 The input matrices A, B, Q , and R are the same as in Example 13.5.8.

$$\text{Step 1. } S = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\text{Step 2. } X_0 = \begin{pmatrix} 0.4 & 0.1 & 0.1 \\ 0.1 & 0.3 & 0 \\ 0.1 & 0 & 0.2 \end{pmatrix}$$

$$\text{Step 3. } i = 0 : \Delta_0 = \begin{pmatrix} -0.0248 & -0.0302 & -0.0369 \\ -0.0302 & -0.0426 & 0.0103 \\ -0.0369 & 0.0103 & -0.0224 \end{pmatrix}$$

$$\alpha_0 = 0.1761, \beta_0 = -0.0049, \gamma_0 = 2.1827 \times 10^{-4}, t_0 = 1.0286$$

$$X_1 = X_0 + t_0 \Delta_0 = \begin{pmatrix} 0.3745 & 0.0690 & 0.0620 \\ 0.0690 & 0.2562 & 0.0105 \\ 0.0620 & 0.0105 & 0.1770 \end{pmatrix}$$

Relative Change: $\| X_1 - X_0 \| / \| X_0 \| = 0.1507$

$$i = 1: \Delta_1 = \begin{pmatrix} -0.0012 & -0.0006 & 0.0000 \\ -0.0006 & 0.0001 & -0.0011 \\ 0.0000 & -0.0011 & 0.0001 \end{pmatrix}$$

$$\alpha_1 = 8.9482 \times 10^{-5}, \beta_1 = -4.2495 \times 10^{-8}, \gamma_1 = 4.9519 \times 10^{-11}, t_1 = 1.0005.$$

$$X_2 = X_1 + t_1 \Delta_1 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\| X_2 - X_1 \| / \| X_1 \| = 0.0038587$

$$i = 2: \Delta_2 = 10^{-6} \begin{pmatrix} -0.1677 & -0.4428 & -0.4062 \\ -0.4428 & -0.7620 & 0.1277 \\ -0.4062 & 0.1277 & -0.2505 \end{pmatrix}$$

$$\alpha_2 = -2.9393 \times 10^{-10}, \beta_2 = -1.0425 \times 10^{-17}, \gamma_2 = 6.1179 \times 10^{-24}, t_2 = 1.0000.$$

$$X_3 = X_2 + t_2 \Delta_2 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\| X_3 - X_2 \| / \| X_2 \| = 2.4025 \times 10^{-6}$

$$i = 3: \Delta_3 = 10^{-12} \begin{pmatrix} -0.1593 & -0.0972 & 0.0319 \\ -0.0972 & -0.0286 & -0.1791 \\ 0.00319 & -0.1791 & 0.0308 \end{pmatrix}$$

$$\alpha_3 = 2.4210 \times 10^{-24}, \beta_3 = -1.4550 \times 10^{-37}, \gamma_3 = 2.4612 \times 10^{-50}, t_3 = 1.0000.$$

$$X_4 = X_3 + t_3 \Delta_3 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\| X_4 - X_3 \| / \| X_3 \| = 5.5392 \times 10^{-13}.$

Theorem 13.5.10 (Convergence of Newton's Method with Line Search for the CARE)

If (A, B) is a stabilizable pair, and if the step sizes t_i are bounded away from zero, then Newton's method with the line search (**Algorithm 13.5.9**) converges to the stabilizing solution.

■

Proof: See Benner and Byers (1998), Guo and Laub (2000).

Flop-count Algorithm 13.5.9 is slightly more expensive (about 8% to the cost of a Newton step) than Algorithm 13.5.8. However, one saves about one iteration step out of 15; often much more, but seldom less.

MATCONTROL Note. Algorithm 13.5.9 has been implemented in MATCONTROL function **ricnwslsc**.

Newton's method for the Discrete Algebraic Riccati Equation

Newton's method for the DARE

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0,$$

is analogous. It is based on successive solutions of **Stein equations (discrete-time Lyapunov equations)** associated with the discrete-time system. We state the algorithm below without detailed discussions. The algorithm was originally developed by Hewer (1971). See also Kleinman (1974).

Algorithm 13.5.10 Newton's Method for the DARE

- Inputs:**
 - A - The $n \times n$ state matrix
 - B - The $n \times m$ input matrix
 - Q - The $n \times n$ state weighting matrix
 - R - The $m \times m$ control weighting matrix

Output: The set $\{X_k\}$ converging to the unique symmetric positive semidefinite stabilizing solution X of the DARE:

$$R_D(X) = A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$

- Assumptions:**
 - 1) (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable.
 - 2) $Q \geq 0, R > 0$.

Step 1. Choose $X_0 = X_0^T$ such that $A - B(R + B^T X_0 B)^{-1} B^T X_0 A$ is a discrete-stable matrix; that is, it has all its eigenvalues inside the unit circle.

Step 2. For $i = 0, 1, 2, \dots$ do until convergence.

- 2.1 Compute $K_i = (R + B^T X_i B)^{-1} B^T X_i A$
 - 2.2 Compute $A_i = A - BK_i$
 - 2.3 Compute $R_D(X_i) = A^T X_i A - X_i + Q - A^T X_i B(R + B^T X_i B)^{-1} B^T X_i A$
 - 2.4 Solve the discrete-time Lyapunov equation (Stein equation) for Δ_i :

$$A_i^T \Delta_i A_i - \Delta_i + R_D(X_i) = 0$$
 - 2.5 Compute $X_{i+1} = X_i + \Delta_i$
- End

The following theorem gives conditions under which the sequence $\{X_i\}$ converges. The proof of this theorem can be found in Lancaster and Rodman (1995, pp. 308-310.)

Theorem 13.5.11 (Convergence of Newton's Method for the DARE)

Suppose that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Let X_0 be a stabilizing approximate solution of the DARE. Then the matrices A_i and X_i , constructed by the above algorithm, are such that

- (i) All A_i are discrete-stable.
- (ii) $0 \leq X \leq \dots \leq X_{i+1} \leq X_i \leq \dots \leq X_1$
- (iii) $\lim_{i \rightarrow \infty} X_i = X$, where X is the unique symmetric positive-semidefinite discrete-stabilizing solution of the DARE.
- (iv) There exists a constant $c > 0$ such that $\|X_{i+1} - X\| \leq c \|X_i - X\|^2$, $i \geq 1$, that is, the sequence $\{X_i\}$ converges quadratically.

Stopping Criterion

The same stopping criteria as in the case of Newton's method for the CARE can be used.

Example 13.5.10 Consider solving the DARE using Algorithm 13.5.10 with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad R = 1, \quad Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Step 1. $X_0 = \begin{pmatrix} 1 & -5 & 10 \\ -5 & 1600 & -2000 \\ 10 & -2000 & 2700 \end{pmatrix}$

Step 2. $i = 0$

The eigenvalues of $A - B(R + B^T X_0 B)^{-1} B^T X_0 A$ are $-0.8831 \pm j0.2910, -0.0222$. Then X_0 is a discrete-stabilizing approximate solution of the DARE.

$$K_0 = \begin{pmatrix} -0.0192 & 2.6154 & -6.8077 \end{pmatrix}$$

$$A_0 = \begin{pmatrix} -0.9808 & -1.6154 & 7.8077 \\ 0.0192 & -4.6154 & 6.8077 \\ 0.0192 & -2.6154 & 3.8077 \end{pmatrix}$$

$$X_1 = 10^4 \begin{pmatrix} 0.0008 & -0.0137 & 0.0167 \\ -0.0137 & 0.6808 & -0.9486 \\ 0.0165 & -0.9486 & 1.3364 \end{pmatrix}$$

Relative Change: $\frac{\| X_1 - X_0 \|}{\| X_0 \|} = 3.7654$

$$i = 1$$

$$K_1 = \begin{pmatrix} -0.0301 & 4.4699 & -9.5368 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} -0.9699 & -3.4699 & 10.5368 \\ 0.0301 & -6.4699 & 9.5368 \\ 0.0301 & -4.4699 & 6.5368 \end{pmatrix}$$

$$X_2 = 10^3 \begin{pmatrix} 0.0067 & -0.0893 & 0.1029 \\ -0.0893 & 2.0297 & -2.5658 \\ 0.1029 & -2.5658 & 3.3125 \end{pmatrix}$$

Relative Change: $\frac{\| X_2 - X_1 \|}{\| X_1 \|} = 0.7364$

$$i = 2$$

$$K_2 = \begin{pmatrix} -0.0826 & 5.1737 & -10.2938 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} -0.9174 & -4.1737 & 11.2938 \\ 0.0826 & -7.1737 & 10.2938 \\ 0.0826 & -5.1737 & 7.2938 \end{pmatrix}$$

$$X_3 = 10^3 \begin{pmatrix} 0.0054 & -0.0670 & 0.0767 \\ -0.0670 & 1.6234 & -2.0796 \\ 0.0767 & -2.0796 & 2.7283 \end{pmatrix}$$

Relative Change: $\frac{\| X_3 - X_2 \|}{\| X_2 \|} = 0.1862.$

The relative changes continue to decrease from this step onwards.

$$X_7 = 10^3 \begin{bmatrix} 0.0053 & -0.0658 & -0.0751 \\ -0.0658 & 1.5943 & -2.0428 \\ 0.0751 & -2.0428 & 2.6817 \end{bmatrix}$$

For $i = 6$, the **relative change**: $\frac{\| X_7 - X_6 \|}{\| X_6 \|}$ is 2.3723×10^{-15} .

MATCONTROL Note. Algorithm 13.5.10 has been implemented in MATCONTROL function **ricnwtnd**.

Newton's Method with Line Search for the DARE

Algorithm 13.5.10 can be modified in a similar way as in case of the CARE to include the line search.

The function $f_i(t)$ to be minimized in this case is given by

$$f_i(t) = \alpha_i(1-t)^2 - 2\beta_i(1-t)t^2 + \gamma_i t^4,$$

where $\alpha_i = \text{Trace}(R_d(X_i)^2)$

$\beta_i = \text{Trace}(R_d(X_i)V_i)$

$\gamma_i = \text{Trace}(V_i^2)$

and $V_i = A_i^T \Delta_i B (R + B^T X_i B)^{-1} B^T \Delta_i A_i$

For details, see Benner (1997), Benner and Byers (1998).

Algorithm 13.5.11 Newton's Method with Line Search for the DARE

Inputs: Same as in Algorithm 13.5.10

Output: Same as in Algorithm 13.5.10

Assumptions: Same as in Algorithm 13.5.10

Step 1. Same as in Algorithm 13.5.10

Step 2. For $k = 0, 1, 2, \dots$ do until convergence

2.1 Same as in Algorithm 13.5.10

2.2 Same as in Algorithm 13.5.10

2.3 Same as in Algorithm 13.5.10

2.4 Same as in Algorithm 13.5.10

2.5 Compute $S_i = B(R + B^T X_i B)^{-1} B^T$

2.6 Compute $V_i = A_i^T \Delta_i S_i \Delta_i A_i$

2.7 Compute the coefficients α_i, β_i and γ_i of $f_i(t)$ as above

2.8 Compute $t_i \in [0, 2]$ such that $f_i(t_i) = \min_{t \in [0, 2]} f_i(t)$.

2.9 $X_{i+1} = X_i + t_i \Delta_i$.

End

Flop-Count The algorithm is again just slightly more expensive than Algorithm 13.5.10. The additional cost of forming V_i , the coefficients of f_i , a local minimizer t_i of f_i and scaling Δ_i by t_i is cheap as compared to $O(n^3)$ flops required for other computations.

Convergence: The line search procedure can sometimes significantly improve the convergence behavior of Newton's method. For details, see Benner (1997).

Example 13.5.11 Consider solving the DARE using Algorithm 13.5.11 with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad R = 1.$$

$$\text{Step 1: } X_0 = \begin{pmatrix} 1 & -5 & 10 \\ -5 & 1600 & -2000 \\ 10 & -2000 & 2700 \end{pmatrix}.$$

$$\text{Step 2: } i = 0, \quad \Delta_0 = 10^4 \begin{pmatrix} 0.0007 & -0.0132 & 0.0157 \\ -0.0132 & 0.5208 & -0.7486 \\ 0.0157 & -0.7486 & 1.0664 \end{pmatrix},$$

$$\alpha_0 = 9.7240 \times 10^7, \quad \beta_0 = 5.5267 \times 10^8, \quad \gamma_0 = 3.1518 \times 10^9, \quad t_0 = 0.3402.$$

$$X_1 = X_0 + t_0 D_0 = 10^3 \begin{pmatrix} 0.0034 & -0.0500 & 0.0635 \\ -0.0500 & 3.3718 & -4.5471 \\ 0.0635 & -4.5471 & 6.3283 \end{pmatrix}$$

$$\text{Relative Change: } \|X_1 - X_0\| / \|X_0\| = 1.2812.$$

$$\text{Step 3: } i = 1, \quad \Delta_1 = 10^3 \begin{pmatrix} 0.0029 & -0.0405 & 0.0431 \\ -0.0405 & -1.1655 & 1.7233 \\ 0.0431 & 1.7233 & -2.6498 \end{pmatrix},$$

$$\alpha_1 = 1.1123 \times 10^7, \quad \beta_1 = 1.7963 \times 10^6, \quad \gamma_1 = 3.0428 \times 10^5, \quad t_1 = 0.8750.$$

$$X_2 = X_1 + t_1 \Delta_1 = 10^3 \begin{pmatrix} 0.0059 & -0.0854 & 0.1012 \\ -0.0854 & 2.3520 & -3.0392 \\ 0.1012 & -3.0392 & 4.0097 \end{pmatrix}$$

$$\text{Relative Change: } \|X_2 - X_1\| / \|X_1\| = 0.3438.$$

$$i = 2, \quad \Delta_2 = 10^{-3} \begin{pmatrix} -0.0006 & 0.0196 & -0.0261 \\ 0.0196 & -0.7570 & 0.9955 \\ -0.0261 & 0.9955 & -1.3267 \end{pmatrix},$$

$$\alpha_2 = 1.9251 \times 10^5, \beta_2 = -157.2798, \gamma_2 = 0.1551, t_2 = 1.0008$$

$$X_3 = X_2 + t_2 \Delta_2 = 10^3 \begin{pmatrix} 0.0053 & -0.0658 & 0.0751 \\ -0.0658 & 1.5944 & -2.0429 \\ 0.0751 & -2.0429 & 2.6819 \end{pmatrix}$$

$$\textbf{Relative Change: } \| X_3 - X_2 \| / \| X_2 \| = 0.3283$$

$$i = 3, \Delta_3 = \begin{pmatrix} -0.0003 & 0.0024 & -0.0011 \\ 0.0024 & -0.0481 & 0.1094 \\ -0.0011 & 0.1094 & -0.2202 \end{pmatrix},$$

$$\alpha_3 = 0.0912, \beta_3 = -2.8785 \times 10^{-5}, \gamma_3 = 1.6525 \times 10^{-8}, t_3 = 1.0003$$

$$X_4 = X_3 + t_3 \Delta_3 = 10^3 \begin{pmatrix} 0.0053 & -0.0658 & -0.0751 \\ -0.0658 & 1.5943 & -2.0428 \\ 0.0751 & -2.0428 & 2.6817 \end{pmatrix}$$

$$\textbf{Relative Change: } \| X_4 - X_3 \| / \| X_3 \| = 6.4273 \times 10^{-5}.$$

$$i = 4, \Delta_4 = 10^{-4} \begin{pmatrix} 0.0001 & 0.0182 & -0.0295 \\ 0.0182 & -0.5017 & 0.4913 \\ -0.0295 & 0.4913 & -0.3757 \end{pmatrix},$$

$$\alpha_4 = 7.4477 \times 10^{-9}, \beta_4 = -1.2874 \times 10^{-15}, \gamma_4 = 4.9961 \times 10^{-22}, t_4 = 1.0000$$

$$X_5 = X_4 + t_4 \Delta_4 = 10^3 \begin{pmatrix} 0.0053 & -0.0658 & 0.0751 \\ -0.0658 & 1.5943 & -2.0428 \\ 0.0751 & -2.0428 & 2.6817 \end{pmatrix}$$

$$\textbf{Relative Change: } \| X_5 - X_4 \| / \| X_4 \| = 2.1982 \times 10^{-8}.$$

$$i = 5, \Delta_5 = 10^{-9} \begin{pmatrix} -0.0001 & 0.0033 & -0.0042 \\ 0.0033 & -0.1537 & 0.2147 \\ -0.0043 & 0.2185 & -0.3126 \end{pmatrix},$$

$$\alpha_5 = 3.6928 \times 10^{-22}, \beta_5 = -1.4449 \times 10^{-34}, \gamma_5 = 2.6879 \times 10^{-46}, t_5 = 1.0000$$

$$X_6 = X_5 + t_5 \Delta_5 = 10^3 \begin{pmatrix} 0.0053 & -0.0658 & 0.0751 \\ -0.0658 & 1.5943 & -2.0428 \\ 0.0751 & -2.0428 & 2.6817 \end{pmatrix}$$

$$\textbf{Relative Change: } \| X_6 - X_5 \| / \| X_5 \| = 1.0906 \times 10^{-13}$$

$$\textbf{Relative Residual} = 3.2312 \times 10^{-11}.$$

MATCONTROL Note: Algorithm 13.5.11 has been implemented in MATCONTROL function `ricnwlsd`.

Newton's Method as an Iterative Refinement Technique

Newton's method is often used as an **iterative refinement technique**. First, a direct robust method such as the Schur method or the matrix sign function method is applied to obtain an approximate solution and this approximate solution is then refined using a few iterative steps of Newton's method. **For higher efficiency, Newton's method with the line search (Algorithm 13.5.9 for the CARE and Algorithm 13.5.11 for the DARE) should be preferred over Newton's method.**

13.6 The Schur and Inverse-Free Generalized Schur Methods for the Descriptor Riccati Equations

As we have seen in Chapter 5 that several practical applications give rise to the descriptor systems:

$$E\dot{x}(t) = Ax(t) + Bu(t) \quad (\text{Continuous-time}) \quad (13.6.1)$$

$$Ex_{k+1} = Ax_k + Bu_k \quad (\text{Discrete-time}) \quad (13.6.2)$$

If E is nonsingular, then the algebraic Riccati equations **associated** with the above systems, respectively, are:

$$A^T X E + E^T X A - E^T X B R^{-1} B^T X E + Q = 0, \quad (13.6.3)$$

and

$$E^T X E = A^T X A - A^T X B (B^T X B + R)^{-1} B^T X A + Q. \quad (13.6.4)$$

The Riccati equations (13.6.3) and (13.6.4) will be, respectively, called as the **descriptor continuous-time algebraic Riccati equation** (DCARE) and the **descriptor discrete-time algebraic Riccati equation** (DDARE).

Most of the methods, such as the Schur method, the matrix sign function method, and Newton's method, can be easily extended to solve DCARE and DDARE.

Below we state how the generalized Schur methods and the inverse-free generalized Schur methods can be extended to solve these equations. The derivations of the others are left as **Exercises**. See Bender and Laub (1985, 1987), Benner (1997), Laub (1991), Mehrmann (1988), etc. in this context.

13.6.1 The Generalized Schur Method for the DCARE

The matrix pencil associated with the DCARE is

$$P_{DCARE} - \lambda N_{DCARE} = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} - \lambda \begin{pmatrix} E & O \\ O & E^T \end{pmatrix},$$

where $S = BR^{-1}B^T$.

The Schur method for the DCARE, then, can be easily developed by transforming the above pencil to the **Ordered real Schur** form using the QZ iteration algorithm (Chapter 4). Thus, if Q_1 and Z_1 are orthogonal matrices such that

$$Q_1 P_{DCARE} Z_1 = \begin{pmatrix} L_{11} & L_{12} \\ O & L_{22} \end{pmatrix}, \quad Q_1 N_{DCARE} Z_1 = \begin{pmatrix} N_{11} & N_{12} \\ O & N_{22} \end{pmatrix},$$

where $Q_1 P_{DCARE} Z_1$ is upper quasi triangular, $Q_1 N_{DCARE} Z_1$ is upper triangular, and $L_{11} - \lambda N_{11}$ is stable; then the columns of $\begin{pmatrix} Z_{11} \\ Z_{21} \end{pmatrix}$, where $Z_1 = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$, span the stable deflating subspace. So, the matrix $X = Z_{21} Z_{11}^{-1}$ is a solution of the DCARE.

MATLAB Note. MATLAB function **care** in the form

$$[X, L, G, rr] = \text{care}(A, B, Q, R, E)$$

solves the DCARE.

Here $G = R^{-1}(B^T X E)$, the gain matrix, $L = \text{eig}(A - BG, E)$ and rr = the Frobenius norm of the relative residual matrix.

13.6.2 The Inverse-Free Generalized Schur Method for the DCARE

In case R is singular or nearly singular, one needs to use the inverse-free generalized Schur method. The extended pencil to be considered in this case is

$$\begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix} - \lambda \begin{pmatrix} E & 0 & 0 \\ 0 & E^T & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

This extended pencil is then compressed into an $2n \times 2n$ pencil in the same way as in Algorithm 13.5.3 and the rest of the procedure is the same as that algorithm.

13.6.3 The Inverse-Free Generalized Schur Method for the DDARE

The matrix pencil associated with the DDARE is

$$\begin{pmatrix} A & 0 \\ -Q & E^T \end{pmatrix} - \lambda \begin{pmatrix} E & S \\ 0 & A^T \end{pmatrix}, \text{ where } S = BR^{-1}B^T.$$

The extended pencil for the **Inverse-free generalized Schur method for the DDARE** is

$$\begin{pmatrix} A & 0 & -B \\ -Q & E^T & 0 \\ 0 & 0 & R \end{pmatrix} - \lambda \begin{pmatrix} E & 0 & 0 \\ 0 & A^T & 0 \\ 0 & B^T & 0 \end{pmatrix}.$$

The pencil is now compressed into an $2n \times 2n$ pencil as in Algorithm 13.5.4 and the rest of the steps of Algorithm 13.5.4 is then followed.

MATLAB Note. The MATLAB function **dare** in the form $[X, L, G, rr] = \text{dare}(A, B, Q, R, E)$ solves the DDARE. Here $G = (B^T X B + R)^{-1} B^T X A$, $L = \text{eig}(A - BG, E)$, and rr = the Frobenius norm of the relative residual matrix.

13.7 Conclusions and Table of Comparisons

In this section, we present a table of comparisons of different methods discussed in this chapter and give a guideline for practical uses of these methods, based on this comparative study. We only present the table below for the CARE. A similar table can be set up for the DARE as well. However, the comments made about the Schur method for the CARE are not valid for the DARE; because *the Schur method for the DARE does not work when A is singular and is expected to give inaccurate results when A is theoretically nonsingular, but is computationally nearly singular.*

A Table of Comparisons of Different Methods for the CARE

Method	Efficiency, Stability, and Convergence Properties	Remarks
The Eigenvector and the Generalized Eigenvector Methods	The methods are in general not numerically stable (They become unstable when the Hamiltonian matrix has nearly multiple eigenvalues).	Not recommended to be used in practice.
The Schur Method	Stable in practice.	Widely used.
The Symplectic Hamiltonian-Schur Method	Stable and Structure-Preserving. Requires less computations and storage for problems of size greater than 20.	Works in the single-input and/or single-output case.
The Extended Hamiltonian-Schur Method	Stable and Structure-Preserving. More-efficient than the Schur-method.	Works in the multi-input case.
Newton's Method	Convergence is quadratic if the initial approximation is close to the solution. The convergence can be painfully slow if the initial approximation is far from the solution.	Usually used as an iterative refinement procedure .
The Matrix Sign-Function Method	Not Stable in general. Though iterative in nature; unlike Newton's Method, does not require the knowledge of a stabilizing initial guess.	Simple to use and is structure preserving. Recommended to be used in conjunction with Newton's method.
The Generalized Schur Method	Stable in practice.	Does not work if the control weighting matrix R is singular. Even if R is theoretically nonsingular, the method should not be used if it is ill-conditioned.
The Inverse-Free Generalized Schur Method	Stable in practice	The best way to solve the CARE when R is nearly singular.

Conclusions and Recommendations: In conclusion, the following recommendations are made: **For the CARE:** *The Schur method (**Algorithm 13.5.1**) or the matrix sign function (**Algorithm 13.5.6**) method followed by Newton's iteration with line search (**Algorithm 13.5.9**) is recommended. If R is singular or nearly singular, then the inverse-free generalized Schur method (**Algorithm 13.5.3**) should be used in place of the Schur method or the matrix sign function method.*

For the DARE: *The inverse-free generalized Schur method (**Algorithm 13.5.4**) or the matrix sign function method (**Algorithm 13.5.7**) followed by Newton's method with line search (**Algorithm 13.5.10**) is recommended. However, the matrix sign function method should be avoided if R is nearly singular.*

13.8 Some Selected Software

13.8.1 MATLAB CONTROL SYSTEM TOOLBOX

Matrix equation solvers.

- care - Solve continuous algebraic Riccati equations
- dare - Solve discrete algebraic Riccati equations

13.8.2 MATCONTROL

- | | |
|----------|---|
| RICEIGC | - The eigenvector method for the continuous-time Riccati equation |
| RICSCHC | - The Schur method for the continuous-time Riccati equation |
| RICSCHD | - The Schur method for the discrete-time Riccati equation |
| RICGEIGD | - The generalized eigenvector method for the discrete-time Riccati equation |
| RICNWTNC | - Newton's method for the continuous-time Riccati equation |
| RICNWTND | - Newton's method for the discrete-time Riccati equatioin |
| RICSGNC | - The matrix sign-function method for the continuous-time Riccati equation |
| RICSGND | - The matrix sign-function method for the discrete-time Riccati equation |
| RICNWLS | - Newton's method with line search for the continuous-time Riccati equation |
| RICNWLD | - Newton's method with line search for the discrete-time Riccati equation |

13.8.3 CSP-ANM

Solutions of the algebraic Riccati equations

- The Schur method is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → SchurDecomposition]` (continuous-time case) and `DiscreteRiccatiSolve [a, b, q, r, SolveMethod → SchurDecomposition]` (**discrete-time case**).
- Newton's method is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → Newton, InitialGuess → w0]` (discrete-time case).
- The matrix Sign-function method is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → MatrixSign]` (continuous-time case) and `DiscreteRiccatiSolve [a, b, q, r, SolveMethod → MatrixSign]` (discrete-time case).
- The inverse-free method based on generalized eigenvectors is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → GeneralizedEigendecomposition]` (continuous-time case) and `DiscreteRiccatiSolve [a, b, q, r, SolveMethod → GeneralizedEigendecomposition]` (discrete-time case).
- The inverse-free method based on generalized Schur decomposition is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → GeneralizedSchurDecomposition]` (continuous-time case) and `DiscreteRiccatiSolve [a, b, q, r, SolveMethod → GeneralizedSchurDecomposition]` (discrete-time case).

13.8.4 SLICOT

Riccati Equations

- SB02MD Solution of algebraic Riccati equations (Schur vectors method)
- SB02MT Conversion of problems with coupling terms to standard problems
- SB02ND Optimal state feedback matrix for an optimal control problem
- SB02OD Solution of algebraic Riccati equations (generalized Schur method)
- SB02PD Solution of continuous algebraic Riccati equations (matrix sign function method) with condition and forward error bound estimates
- SB02QD Condition and forward error for continuous Riccati equation solution
- SB02RD Solution of algebraic Riccati equations (refined Schur vectors method) with condition and forward error bound estimates
- SB02SD Condition and forward error for discrete Riccati equation solution

13.8.5 MATRIX_X

Purpose: Solve Riccati equation. Using the option ‘DISC’ solves the discrete Riccati equation.

Syntax: [EV, KC]=RICCATI (S, Q, NS, ‘DISC’)
[EV, KC, P]=RICCATI (S, Q, NS, ‘DISC’)

Purpose: Solves the indefinite Algebraic Riccati Equation (ARE): $A'P + PA - PRP + Q = 0$

Syntax: [P, SOLSTAT]=SINGRICKATI (A, Q, R { ,TYPE})

13.9 Summary and Review

As we have seen in Chapter 10 and Chapter 12 that the algebraic Riccati equations

$$XA + A^T X - XBR^{-1}B^T X + Q = 0 \text{ (CARE)}$$

and

$$A^T XA - X + Q - A^T XB(R + B^T XB)^{-1}B^T XA = 0 \text{ (DARE)}$$

arise in many areas of control systems design and analysis, such as:

- The LQR and LQG Designs
- Optimal State Estimation (Kalman Filter)
- H_∞ -Control
- Spectral Factorizations (not described in this book, see Van Dooren (1981)).

I. Existence and Uniqueness of Stabilizing Solution. Let $Q \geq 0$ and $R > 0$. If (A, B) is stabilizable and (A, Q) is detectable, then the CARE admits a unique symmetric positive semidefinite stabilizing solution (**Theorem 13.2.6**).

Such a solution is given by $X = X_2 X_1^{-1}$, where the columns of the matrix $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ span the stable invariant subspace of the **Hamiltonian matrix** $H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}$, where $S = BR^{-1}B^T$.

An analogous result for the DARE also exists (**Theorem 13.3.2**). In this case the symplectic matrix

$$M = \begin{pmatrix} A + S(A^{-1})^T Q & -S(A^{-1})^T \\ (-A^{-1})^T Q & (A^{-1})^T \end{pmatrix}$$

takes the role of the Hamiltonian matrix.

II. Conditioning of the Riccati Equations

The absolute and the relative condition numbers of the CARE have been identified using a perturbation result (**Theorem 13.4.1**).

An approximate condition number of the CARE, using a first-order estimate is Byers' condition number (in **Frobenius norm**):

$$\kappa_{CARE}^B = \frac{\|\Omega^{-1}\| \|Q\| + \|\Theta\| \|A\| + \|\Pi\| \|S\|}{\|X\|},$$

where X is the stabilizing solution of the CARE and Ω , Π , and Θ are defined by:

$$\begin{aligned} \Omega(Z) &= (A - SX)^T Z + Z(A - SX), \\ \Theta(Z) &= \Omega^{-1}(Z^T X + XZ), \\ \Pi(Z) &= \Omega^{-1}(XZX), \\ \|\Omega^{-1}\|_F &= \frac{1}{\text{sep}(A_C^T, -A_C)}, \end{aligned}$$

where $A_C = A - SX$, $S = BR^{-1}B^T$.

The quantities $\|\Omega^{-1}\|$, $\|\Theta\|$, and $\|\Pi\|$ are computationally intensive. However lower and upper bounds of κ_{CARE}^B can be obtained by solving the following Lyapunov equations:

$$(A - SX)^T H_k + H_k(A - SX) = -X^k, k = 0, 1, 2$$

The large norms of these matrices (relative to the stabilizing solution X), in general, indicate that the CARE is ill-conditioned.

The condition number of the DARE is given by (13.4.16).

A first-order estimator for the condition number of the DARE is

$$\kappa_{DARE}^E = \frac{2 \| A \|_F^2 \frac{\| Q \|_F}{\| X \|_F} + \| A \|_F^2 \| S \|_F \| X \|_F}{sep(A_d^T, A_d)},$$

where $A_d = A - B(R + B^T X B)^{-1} B^T X A$, $S = BR^{-1}B^T$. The quantity $sep(A_d^T, A_d)$ can be determined as the minimum singular value of the matrix $A_d^T \otimes A_d^T - I_n^2$.

III. Numerical Methods for the Riccati Equations

The existing numerical methods for the Riccati equations can be broadly classified into the following four classes:

- Invariant Subspace Methods
- Deflating Subspace Methods
- The Matrix Sign Function Methods
- Newton's Methods

A basic idea of finding a stabilizing solution of the CARE (DARE), using an invariant subspace method, is to construct a basis for the stable invariant subspace of the Hamiltonian matrix H (symplectic matrix M). Such a basis can be constructed by using the eigenvectors or the Schur vectors of the Hamiltonian matrix H (the symplectic matrix M). The eigenvector matrix can be ill-conditioned if the matrix H (the matrix M) is nearly defective; and, therefore, **the eigenvector approach is not recommended to be used in practice.** The Schur method is preferable to the eigenvector method. If $U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$ is the **ordered Real Schur form** of H , and the eigenvalues with negative real parts are contained in T_{11} , then $X = U_{21} U_{11}^{-1}$ is the stabilizing solution of the CARE, where $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$.

The Schur method for the DARE can be similarly developed by finding an ordered real Schur form of the symplectic matrix M . However, since computation of the matrix M requires the explicit inversion of A , **the Schur method for the DARE does not work if A is singular or can be problematic if A is theoretically nonsingular but is computationally singular.** In such cases, a deflating subspace method should be used.

The idea behind a deflating subspace method is basically the same as that of an invariant subspace method except that the solution of the Riccati equation is now found by computing a basis for the deflating subspace of a matrix pencil. For the CARE, the pencil in $P_{CARE} - \lambda N_{CARE}$, where $P_{CARE} = \begin{pmatrix} A & -S \\ -Q & A^T \end{pmatrix}$, $N_{CARE} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$. For the

DARE, the matrices of the pencil are

$$P_{DARE} = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}, \quad N_{DARE} = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}.$$

Again, for reasons stated above, the **generalized Schur decomposition using the QZ algorithm should be used to compute such a basis. See Section 13.5.2 for details. The eigenvector approach should be avoided.**

Both the Schur methods and the generalized Schur methods require an explicit inversion of the matrix R . In case R is ill-conditioned with respect to matrix inversion, these methods may not give accurate solutions. The difficulties can be overcome by using an extended $(2n + m) \times (2n + m)$ pencil.

For the CARE, the extended pencil is $P_{CARE}^E - \lambda N_{CARE}^E$, where

$$P_{CARE}^E = \begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix},$$

and

$$N_{CARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

This extended $(2n + m) \times (2n + m)$ pencil can then be compressed into a $2n \times 2n$ pencil by finding the QR factorization of $\begin{pmatrix} R \\ B \end{pmatrix}$, without affecting the deflating subspace. The solution of the CARE then can be obtained by finding the ordered generalized Schur form of the compressed pencil.

For the DARE, the extended pencil is $M_{DARE}^E - \lambda N_{DARE}^E$, where

$$P_{DARE}^E = \begin{pmatrix} A & 0 & -B \\ -Q & I & 0 \\ 0 & 0 & R \end{pmatrix},$$

and

$$N_{DARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & A^T & 0 \\ 0 & B^T & 0 \end{pmatrix}.$$

This $(2n + m) \times (2n + m)$ can be compressed into an $2n \times 2n$ pencil by using the QR factorization of $\begin{pmatrix} R \\ -B \end{pmatrix}$. For details, see **Section 13.5.2**.

Again, the required basis should be constructed by finding the generalized real Schur form of the pencil using the QZ algorithm.

13.10 Chapter Notes and Further Reading

The algebraic Riccati equations have been very well studied in the literatures of mathematics and control and filter theory.

For an excellent account of up-to-date theoretical developments, see the recent book of Lancaster and Rodman (1995). Some of the earlier theoretical developments are contained in Kučera (1972, 1979), Coppel (1974), and Singer and Hammarling (1983), Willems (1971), Wimmer (1984, 1994). The books by Anderson and Moore (1990), Ando (1988), Kwakernaak and Sivan (1972), Kimura (1997), Zhou et al. (1996) also contain a fair amount of theory of algebraic Riccati equations. The existence of maximal solutions for generalized algebraic Riccati equations arising in stochastic control has been discussed in DeSouza and Fragoso (1990). The paper by DeSouza, Gevers and Goodwin (1986) deals with Riccati equations arising in optimal filtering of nonstabilizable systems having singular state transition matrices.

Important numerical methods have been dealt with in details in the books by Sima (1996) and Mehrmann (1991). Benner (1999) has given an up-to-date review with special attention to structure-preserving methods. An extensive bibliography on numerical methods appear in Laub (1991), and Benner (1997). See Jamshidi (1980) for an earlier review.

For a review of periodic Riccati equations see the article of Bittanti et al. and the references therein in the book “**The Riccati Equation**” edited by Bittanti, Laub and Willems (1991). The latter contains several important papers on Riccati equations and the paper by Bittanti gives a brief life history of Count Jacopo Riccati (1676-1754), which is certainly worth reading. The sensitivity of the continuous-time Riccati equations has been studied by several people: Byers (1985), Kenney and Hewer (1990), Chen (1988), Konstantinov et.al. (1990), Xu (1996), Sun (1998), and Ghavimi and Laub (1995). **Theorem 13.4.1** is due to Sun (1998). The bound (13.4.14) is due to Kenney and Hewer (1990). The residual of an approximate stabilizing solution (**Theorem 13.4.3**) is due to Sun (1997a). The sensitivity of the discrete-time algebraic Riccati equation has been studied in Gudmundsson, Kenney and Laub (1992), Konstantinov et. al. (1993), and Sun (1998). The paper by Ghavimi and Laub (1995) relates backward error and sensitivity to accuracy and discusses techniques for refinement of computed solutions of the algebraic Riccati equations.

The eigenvector methods for the Riccati equations were proposed by McFarlane (1963) and Potter (1966). The Schur method for the Riccati equations originally appeared in the famous paper by Laub (1979). Petkov, Christov, and Konstantinov (1987) studied the numerical properties of the Schur method and concluded that the Schur method can be unstable in some cases and the solutions may be inaccurate. A further analysis by Kenney, Laub, and Wette (1989) attributed such inaccuracy to poor scaling. For an excellent account of scaling of the Schur methods, see Benner (1997). The structure-preserving Hamiltonian-Schur method was first proposed by Byers in his Householder-prize winning Ph.D thesis (1983) in the case of a single-input system ($\text{rank}(B) = 1$). See Byers (1986a) for details of the method. The theoretical foundation of this method is contained in the well-known paper by Paige and Van Loan (1981). Their result was later extended to the case when the Hamiltonian matrix has eigenvalues on the

imaginary axis by Lin and Ho (1990). Patel, Lin and Misra (1994) have discussed computation of stable invariant subspaces of Hamiltonian matrices. Another method, called the multishift method to compute the invariant subspace of the Hamiltonian matrix corresponding to the stable eigenvalues, was developed by Ammar and Mehrman (1993). The algorithm is called multishift algorithm because n stable eigenvalues of the Hamiltonian matrix are used as shifts to isolate the desired invariant subspace. The multishift method sometimes has convergence problems, particularly for large n . The Hamiltonian-Schur algorithm in the multi-input case is due to Benner, Mehrmann and Xu (1997). A good account of the structure preserving eigenvalue methods appears in Bunse-Gerstner, Byers and Mehrmann (1992). Mehrmann (1988) has given a structure-preserving method for the discrete-time Riccati equation with single-input and single-output. The non-orthogonal symplectic methods have been discussed by Bunse-Gerstner and Mehrmann (1986) and Bunse-Gerstner, Mehrmann and Watkins (1989) for the CARE, and by Benner, Fassbender and Watkins (1999), Fassbender and Benner (2001) for the DARE. The details of these methods and other references can be found in the recent book by Fassbender (2000).

The generalized eigenvalue problem approach leading to deflating subspace method for the discrete-time Riccati equation was proposed in Pappas, Laub, and Sandell (1980). See also Arnold and Laub (1984), Emami-Naeini and Franklin (1979, 1982). The inverse-free methods (the extended pencil approach (**Algorithm 13.5.3** and **Algorithm 13.5.4**)) and the associated compressed techniques were proposed by Van Dooren (1981).

The idea of using matrix sign function to solve the CARE was first introduced by Roberts (1971, 1980). Byers (1986b, 1987) discussed numerical stability of the method and studied the computational aspects in details. See also Bierman (1984) and Bai and Demmel (1998). A generalization of the matrix sign function method to a matrix pencil and its application to the solutions of DCARE and DDARE was proposed by Gardiner and Laub (1986). For a summary of the matrix sign function, see the recent paper of Kenney and Laub (1995). For a perturbation analysis of the matrix sign function, see Sun (1997c). Howland (1983) relates matrix sign function to separation of matrix eigenvalues.

For details of Newton's algorithm for the continuous-time algebraic Riccati equation (**Algorithm 13.5.8**) and that for the discrete-time algebraic Riccati equation (**Algorithm 13.5.10**), as presented here, see Benner (1997), Lancaster and Rodman (1995). The correct proof of convergence of Newton's method (**Theorem 13.5.8**) seemed to appear for the first time in Lancaster and Rodman (1995).

Kenney, Laub, and Wette (1990) gave results on error bounds for Newton's method, where it was first pointed out that if the initial solution X_0 is not chosen carefully, the error on the first step can be disastrous. They also gave conditions which guarantee monotone convergence from the first step on (**Theorem 13.5.9**). Several modifications of Newton's methods have appeared in recent years (Guo (1998), Guo and Lancaster (1998), Guo and Laub (2000), etc.). The line search modification proposed by Benner and Byers (1998) is extremely useful in practice. In general, it improves the convergence behavior of Newton's method and avoids the problem of

a disastrously large first step.

Ghavimi, Kenney, and Laub (1992) have discussed the local convergence analysis of conjugate gradient methods for solving the algebraic Riccati equations.

For an account of parallel algorithms for algebraic Riccati equations, see Bai and Qian (1994), Gardiner and Laub (1991), and Laub (1991) and references therein, Quintana and Hernández (1996a, 1996b, 1996c), etc.

For large-scale solutions of the algebraic Riccati equations see Ferng, Lin, and Wang (1997), Lu and Lin (1993), Jaimoukha and Kasenally (1994) and Benner and Fassbender (1997). The recent book by Ionescu, Oara and Weiss (1999) gives a nice treatment of algebraic Riccati equations for the indefinite sign and singular cases. See also Campbell (1980). For least-squares solutions of stationary optimal control using the algebraic Riccati equations, see Willems (1971).

Some discussions on finding the Cholesky factor of the solution to an algebraic Riccati equation without first computing the solution itself appears in Singer and Hammarling (1983). Lin (1987) has given a numerical method for computing the closed-loop eigenvalues of a discrete-time Riccati equation. Patel (1993) has given a numerical method for computing the eigenvalues of a symplectic matrix. For numerical algorithms for descriptor Riccati equations, see Benner (1999), Mehrmann (1991), Bender and Laub (1985, 1987), Benner, Mehrmann and Xu (1999), etc. A description of discrete-time descriptor Riccati equations also appears in Zhang, Lam, and Zhang (1999). A comparative study with respect to efficiency and accuracy of most of the methods described in this chapter for the CARE (the **eigenvector, Schur, inverse-free generalized Schur, Hamiltonian-Schur and Newton's Methods**) has been made in the recent M.Sc Thesis of Ho (2000), using MATLAB and FORTRAN-77 codes (In particular, this thesis contains MATLAB codes for ordered **Real Schur** and **Generalized Real Schur** decompositions). Numerical experiments were performed on 12 benchmark examples taken from the collection of Benner, Laub, and Mehrmann (1995). The conclusions drawn in this thesis are almost identical to those mentioned in **Section 13.7**. For a recent collection of benchmark examples for Riccati equations, see Abels and Benner (1999a, 1999b).

EXERCISES

1. Prove that if (A, B) is controllable and (A, Q) is observable, then the CARE (13.1.1) admits a unique symmetric positive definite solution X .
2. Construct an example to show that the observability of (A, Q) is not necessary for the solution X of the CARE (13.1.1) to be positive definite.
3. Prove that the matrix

$$M = \begin{pmatrix} A + S(A^{-1})^T Q & -S(A^{-1})^T \\ -(A^{-1})^T Q & (A^{-1})^T \end{pmatrix},$$

$S = BR^{-1}B^T$, associated with the DARE:

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A$$

is symplectic, and that if λ is a nonzero eigenvalue of M , so is $\frac{1}{\lambda}$.

4. Establish the relation (13.2.16).
5. (a) Prove the discrete counterpart of Theorem 13.2.4; that is, prove that the DARE (13.1.2) has a unique symmetric positive semidefinite stabilizing solution if and only if (A, B) is discrete-stabilizable and the associated symplectic matrix M does not have an eigenvalue on the unit circle.
- (b) Prove the discrete counterpart of Theorem 13.2.5; that is prove that if (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable, then the symplectic matrix M defined by (13.3.1) does not have an eigenvalue on the unit circle.
- (c) Using the results of Problem 3, and those of 5(a) and 5(b), prove Theorem 13.3.2.
6. Prove that the homogeneous CARE: $XA + A^T X + XSX = 0$ has a stabilizing solution if A has no eigenvalues on the imaginary axis. Prove or disprove a discrete-counterpart of this result.
7. Prove that the quantity

$$\frac{2 \|A\|_F^2 \frac{\|Q\|_F}{\|X\|_F} + \|A\|_F^2 \|S\|_F \|X\|_F}{sep_d(A_d^T, A_d)},$$

where $A_d = A - B(R + B^T X B)^{-1} B^T X A$, serves as an approximate condition number of the DARE (13.1.2). Construct an example of an ill-conditioned DARE using this quantity.

8. Assume that $A - SX$ is asymptotically stable. Let $H_k, k = 0, 1, 2$ satisfy the Lyapunov equations:

$$(A - SX)^T H_k + H_k (A - SX) = -X^k,$$

Then prove that in 2-norm $\left[\frac{\|H_0\| \|H_2\|}{Cond(X)} \right]^{\frac{1}{2}} \leq \|H_1\| \leq \|H_0\|^{\frac{1}{2}} \|H_1\|^{\frac{1}{2}}$, where X is positive definite.

9. Find an example to illustrate that a small relative residual in a computed solution of the CARE does not guarantee a small error in the solution.
10. Prove that if Ω is singular, then $sep((A - SX), -(A - SX)^T)$ is zero.
11. Give a proof of the theoretical basis of Schur algorithm for the CARE using Theorem 13.2.6. That is, prove the following: Let H be the Hamiltonian matrix associated with the CARE and $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$ be the unitary matrix such that $U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$, where the n eigenvalues of H with negative real parts are stacked in T_{11} . Then, $X = U_{21} U_{11}^{-1}$ is the unique positive semidefinite stabilizing solution of the CARE.

12. Construct an example to show that the solution of the CARE, obtained by the Schur method, might be inaccurate, even though the problem is not ill-conditioned. (**Hint:** Construct an example for which U_{11} is ill-conditioned, but the CARE is well-conditioned).
13. Give an example to demonstrate the superiority of the Schur algorithm for the CARE over the eigenvector algorithm, in case the associated Hamiltonian matrix is nearly defective.
14. Using Theorem 13.5.1 and the transformation

$$H = (M + I_{2n})(M - I_{2n})^{-1},$$

prove Theorem 13.5.2.

15. Construct an example to demonstrate the numerical difficulties of the Schur algorithm for the DARE in case the matrix A is nearly singular.
16. Write down an algorithm for solving the discrete algebraic Riccati equation, using the eigenvectors of the symplectic matrix. Discuss the computational drawbacks of the algorithm. Construct an example to illustrate the computational drawbacks.
17. Prove the properties 1 through 5 of the matrix sign-function $\text{Sign}(A)$ stated in Section 13.5.3.
18. Prove that if $|\lambda| = 1$ is an eigenvalue of the pencil $P_{DARE} - \lambda N_{DARE}$ with the eigenvector $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$, where P_{DARE} and N_{DARE} are the same as given in Theorem 13.5.5, then the detectability of (A, Q) implies that $z_1 = 0$.
19. Develop the generalized Schur methods for the CARE and DARE in details.
20. Why is the generalized Schur method not preferable over the Schur method for the CARE if R is not nearly singular?
21. Construct an example to demonstrate the poor accuracy of the generalized eigenvector method for the DARE in case the pencil $P_{DARE} - \lambda N_{DARE}$ has near multiple eigenvalues. Apply the generalized Schur algorithm (Algorithm 13.5.2) to the same example and verify the improvement in the accuracy of the solution.
22. Work out the details of how the pencil $P_{CARE}^E - \lambda N_{CARE}^E$ can be transformed to the compressed pencil $P_{CARE}^{EC} - \lambda N_{CARE}^{EC}$ using the QR factorization of the matrix $\begin{pmatrix} B \\ R \end{pmatrix}$.
23. Repeat the exercise for the DARE; that is, work out the details of the transformation to the pencil $P_{DARE}^{EC} - \lambda N_{DARE}^{EC}$ using the QR factorization of the matrix $\begin{pmatrix} R \\ -B \end{pmatrix}$.

24. Prove that the pencil $P_{CARE}^E - \lambda N_{CARE}^E$ and the pencil $P_{CARE}^{EC} - \lambda N_{CARE}^{EC}$ as defined in Section 13.5.2 for the CARE have the same deflating subspaces; and similarly for the DARE.
25. Develop the following algorithms in detail for both the DCARE and DDARE (consult Laub (1991), Benner (1997)).
- (1) The Schur algorithms
 - (2) The Generalized Schur algorithms
 - (3) The Inverse-free generalized Schur algorithms
 - (4) The Matrix sign-function algorithms
 - (5) Newton's algorithms

Construct a simple example to illustrate each of the above algorithms.

26. Construct an example to demonstrate the superiority of the inverse-free generalized Schur algorithm over the Schur algorithm for the CARE, in case the control weighting matrix R is positive definite but nearly singular.
27. Carry out a numerical experiment with a 150×150 randomly generated problem to make a comparative study with respect to computer-time and accuracy of the solution to the CARE with the following methods: the eigenvector method, the Schur method, inverse-free generalized Schur method, the matrix sign function method and the Hamiltonian structure preserving Schur method. Write down your observations and conclusions.
28. Repeat the previous exercise with the DARE using the following methods: The eigenvector method, the generalized eigenvector method, the Schur method, the generalized Schur method, inverse-free generalized Schur method, and the matrix sign function method.

RESEARCH PROBLEMS ON CHAPTER 13

1. Develop a structure-preserving method to compute the symplectic Schur decomposition and apply the method to solve the DARE; thus obtaining a symplectic structure-preserving method for the DARE.

References

1. J. Abels and P. Benner. CAREX—a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0). *SLICOT Working Note* 1999-14, November 1999a. (Available at the NICONET Website: <http://www.win.tue.ne/niconet/niconet.html>).
2. J. Abels and P. Benner. DAREX—a collection of benchmark examples for discrete-time algebraic Riccati equations (version 2.0). *SLICOT Working Note* 1999-15, November 1999b. (Available at the NICONET Website: <http://www.win.tue.ne/niconet/niconet.html>).
3. G. Ammar and V. Mehrmann, A multishift algorithm for the numerical solution of algebraic Riccati equations, *Electr. Trans. Num. Anal.*, vol. 1, pp. 33-48, 1993.
4. B.D.O. Anderson, and J.B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice Hall, Englewood Cliffs, NJ, 1990.
5. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, 3rd Edition, SIAM, Philadelphia, 1999.
6. T. Ando, *Matrix quadratic equations*, Hokkaido University, Research Institute of Applied Electricity, Division of Applied Mathematics, Sapporo, Japan, 1988.
7. W. Arnold, III and A. Laub, Generalized eigenproblem algorithms and software for algebraic Riccati equations, *Proc. IEEE*, vol. 72, pp. 1746-1754, 1984.
8. Z. Bai and J. Demmel, Using the matrix sign function to compute invariant subspaces, *SIAM J. Matrix Anal. Appl.*, vol. 19 , pp. 205-225, 1998.
9. Z. Bai and Q. Qian, Inverse free parallel method for the numerical solution of algebraic Riccati equations, *Proc. Fifth SIAM Conf. Appl. Lin. Alg.*, Snowbird, UT, June, pp. 167-171, 1994, (J. Lewis, Editor).
10. D. Bender and A. Laub, The linear-quadratic optimal regulator problem for descriptor systems, *Proc. 24th IEEE Conf. Dec. Control*, Ft. Lauderdale, FLorida, December, pp. 957-962, 1985.
11. D. Bender and A. Laub, The linear-quadratic optimal regular for descriptor systems: Discrete-time case, *Automatica*, vol. 23, pp. 71-85, 1987.
12. P. Benner, Computational Methods for Linear-Quadratic Optimization, *Rendiconti del Circulo Matematico di Palermo*, Supplemento, Serie II, no. 58, pp. 21-56, 1999.
13. P. Benner, *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*, Dissertation for Dipl.-Math., Technischen Universität Chemnitz-Zwickau, Germany 1997.

14. P. Benner and R. Byers, An exact line search method for solving generalized continuous-time algebraic Riccati equations, *IEEE Trans. Automat. Control*, pp. 101-107, 1998.
15. P. Benner, R. Byers, V. Mehrmann, and H. Xu, Numerical solution of linear-quadratic control problems for descriptor systems, *Proc. 1999 IEEE Intl. Symp. CACSD*, Kohala Coast-Island of Hawaii, Hawaii, USA, August 22-27, 1999, pp.46-51, 1999, (O. Gonzalez, Editor).
16. P. Benner and H. Fassbender, An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem, *Lin. Alg. Appl.* vol. 263, pp. 75-111, 1997.
17. P. Benner, H. Fassbender, and D. Watkins, Two connections between the SR and HR eigenvalue algorithms, *Lin. Alg. Appl.*, vol. 272, pp.17-32, 1997.
18. P. Benner, H. Fassbender and D. Watkins, SR and SZ algorithms for the symplectic (butterfly) eigenproblem, *Lin. Alg. Appl.*, vol. 287, pp.41-76, 1999.
19. P. Benner, A. Laub, and V. Mehrmann, A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: Continuous-time case, *Tech. Report SPC 95-22, Fak. f. Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, 1995*. (Available at the website: www.math.uni-bremen.de/~benner).
20. P. Benner, A. Laub, and V. Mehrmann, A collection of benchmark examples for the numerical solution of algebraic Riccati equations II: Discrete-time case, *Tech. Report SPC 95-23, Fak. f. Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, 1995*. (Available at the website: www.math.uni-bremen.de/~benner).
21. P. Benner, A. Laub, and V. Mehrmann, Benchmarks for the numerical solution of algebraic Riccati equations. *IEEE Control Systems Magazine*, vol. 7, no. 5, pp. 18-28, 1997.
22. P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga, SLICOT—a subroutine library in systems and control theory. *Applied and Computational Control, Signals, and Circuits*, volume 1, Chapter 10, pp. 499-539. Birkhauser, Boston, MA, 1999. (B.N. Datta, et al., Editors).
23. P. Benner, V. Mehrmann, and H. Xu, A new method for computing the stable invariant subspace of a real Hamiltonian matrix, *J. Comput. Appl. Math.*, vol. 86, pp. 17-43, 1997.
24. P. Benner, V. Mehrmann, and H. Xu, A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils, *Numer. Math.*, vol. 78, pp. 329-358, 1999.
25. G. J. Bierman, Computational aspects of the matrix sign function solution to the ARE, Proc. 23rd *IEEE Conf. Dec. Contr.*, Las Vegas, Nevada, pp. 514-519, 1984.

26. S. Bittanti, A. Laub, and J.C. Willems, (Editors), *The Riccati Equation*, Springer-Verlag, Berlin, 1991.
27. A. Bunse-Gerstner, V. Mehrmann, and D. Watkins, An SR algorithm for Hamiltonian matrices based on Gaussian elimination, *Methods of Operations Research*, vol. 58, pp. 339-358, 1989.
28. A. Bunse-Gerstner, R. Byers, and V. Mehrmann, A chart of numerical methods for structured eigenvalue problems, *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 2, pp. 419-453, 1992.
29. A. Bunse-Gerstner and V. Mehrmann, A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation, *IEEE Trans. Automat. Control*, AC-31, pp. 1104-1113, 1986.
30. R. Byers, *Hamiltonian and Symplectic Algorithms for the Algebraic Riccati Equation*, PhD thesis, Cornell University, Dept. Comp. Sci., Ithaca, NY, 1983.
31. R. Byers, Numerical condition of the algebraic Riccati equation, *Contemp. Math.*, Amer. Math. Soc., Providence, RI, vol. 47, pp. 35-49, 1985, (R. Brualdi, et al., Editor).
32. R. Byers, A Hamiltonian QR-algorithm, *SIAM J. Sci. Statist. Comput.*, vol. 7, pp. 212-229, 1986a.
33. R. Byers, Numerical stability and instability in matrix sign function based algorithms in *Computational and Combinatorial Methods in Systems Theory*, pp. 185-200, North Holland, New York, 1986b, (C. I. Byrnes and A. Lindquist, Editors).
34. R. Byers, Solving the algebraic Riccati equation with the matrix sign function, *Lin. Alg. Appl.*, vol. 85, pp. 267-279, 1987.
35. R. Byers, A Hamiltonian Jacobi Algorithm, *IEEE Trans. Automat. Control*, vol. 35, no. 5, pp. 566-570, 1990.
36. S. L. Campbell, *Singular Systems of Differential Equations*, Pitman, Marshfield, MA, 1980.
37. C.-H. Chen, Perturbation analysis for solutions of algebraic Riccati equations, *J. Comput. Math.*, vol. 6, pp. 336-347, 1988.
38. W. A. Coppel, Matrix quadratic equations, *Bull. Australian Math. Soc.*, vol. 10, pp. 327-401, 1974.
39. E. D. Denman and A. N. Beavers. The matrix sign function and computations in systems, *Appl. Math. Comput.*, vol. 2, pp. 63-94, 1976.

40. C. E. DeSouza and M. D. Fragoso, On the existence of maximal solution for generalized algebraic Riccati equations arising in stochastic control, *Syst. Contr. Lett.*, vol. 14, pp. 223-239, 1990.
41. C. E. DeSouza, M. R. Gevers, and G. C. Goodwin, Riccati equations in optimal filtering of nonstabilizable systems having singular state transition matrices, *IEEE Trans. Automat. Control*, vol. AC-31, pp. 831-838, 1986.
42. J. Doyle, K. Glover, P. Khargonekar, and B. Francis, State-space solutions to standard H_2 and H_∞ control problems, *IEEE Trans. Automat. Control*, AC-34, pp. 831-847, 1989.
43. A. Emami-Naeini and G. F. Franklin, Design of steady state quadratic loss optimal digital controls for systems with a singular system matrix, *Proc. 13th Asilomar Conference Circ. Syst. Comp.*, pp. 370-374, 1979.
44. A. Emami-Naeini and G. F. Franklin, Deadbeat control and tracking of discrete-time systems, *IEEE Trans. Automat. Control*, vol. AC-27, pp. 176-181, 1982.
45. H. Fassbender and P. Benner, A hybrid method for the numerical solution of discrete-time algebraic Riccati equations, *Contemporary Mathematics on Structured Matrices in Mathematics, Computer Science, and Engineering*, Amer. Math. Soc., Providence, RI, vol. 280, pp. 255-269, 2001, (V. Olshevsky , Editor).
46. H. Fassbender, *Symplectic Method for the Symplectic Eigenproblem*, Kluwer Academic/Plenum Publishers, New York, 2000.
47. W. R. Ferng, W.-W. Lin, and C.-S. Wang, The shift-inverted J -Lanczos algorithm for the numerical solutions of large sparse algebraic Riccati equations, *Comput. Math. Appl.*, vol. 33, no. 10, pp. 23-40, 1997.
48. P.M. Gahinet, A.J. Laub, C.S. Kenney, and G.A. Hewer, *Sensitivity of the stable discrete-time Lyapunov equation*, IEEE Trans. Automat. Control, vol. 35, pp. 1209-1217, 1990.
49. J.D. Gardiner, Stabilizing control for second-order models and positive real systems, *AIAA J. Guidance, Dynamics and Control*, vol. 15, pp. 280-282, 1992.
50. J.D. Gardiner and A.J. Laub, A generalization of the matrix-sign-function solution for algebraic Riccati equations, *Int. J. Control*, vol. 44, pp. 823-832, 1986.
51. J.D. Gardiner and A.J. Laub, Parallel algorithms for algebraic Riccati equations, *Int. J. Control*, vol. 54, pp. 1317-1333, 1991.
52. J.D. Gardiner, A.J. Laub, J.J. Amato, and C.B. Moler, Solution of the Sylvester matrix equation $AXB + CXD = E$, *ACM Trans. Math. Software*, vol. 18, pp. 223-231, 1992.

53. A. Ghavimi, C. Kenney, and A.J. Laub, Local convergence analysis of conjugate gradient methods for solving algebraic Riccati equations, *IEEE Trans. Automat. Control*, vol. AC-37, pp. 1062-1067, 1992.
54. A. R. Ghavimi and A.J. Laub, Backward error, sensitivity, and refinement of computed solutions of algebraic Riccati equations, *Num. Lin. Alg. Appl.*, vol. 2, pp. 29-49, 1995.
55. T. Gudmundsson, C. Kenney, and A.J. Laub, Scaling of the discrete-time algebraic Riccati equation to enhance stability of the Schur method, *IEEE Trans. Automat. Control*, AC-37, pp. 513-518, 1992.
56. C.-H. Guo, Newton's method for discrete algebraic Riccati equations when the closed-loop matrix has eigenvalues on the unit circle, *SIAM J. Matrix Anal. Appl.*, vol. 20, pp. 279-294, 1998.
57. C.-H. Guo and P. Lancaster, Analysis and modification of Newton's method for algebraic Riccati equations. *Math. Comp.*, vol. 67, pp. 1089-1105, 1998.
58. C.-H. Guo and A.J. Laub, on a Newton-like method for solving algebraic Riccati equations, *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 694-698, 2000.
59. S.J. Hammarling, Newton's method for solving the algebraic Riccati equation, *NPL Report DITC 12/82, National Physical Laboratory, Teddington*, Middlesex TW11 OLW, U.K., 1982.
60. G.A. Hewer, An iterative technique for the computation of steady state gains for the discrete optimal controller, *IEEE Trans. Automat. Control*, vol. AC-16, pp. 382-384, 1971.
61. B. Hinrichsen, B. Kelb, and A. Linnemann, An algorithm for the computation of the structured complex stability radius, *Automatica*, vol. 25, pp. 771-775, 1989.
62. D. Hinrichsen and A.J. Pritchard, Stability radii of linear systems, *Sys. & Contr. Lett.*, vol. 7, pp. 1-10, 1986.
63. D. Hinrichsen, A.J. Pritchard, and S.B. Townley, Riccati equation approach to maximizing the complex stability radius by state feedback, *Int. J. Control*, vol. 52, pp. 769-794, 1990.
64. T. Ho, *A study of computational methods for the continuous-time algebraic Riccati equation*, M.Sc. Thesis, Northern Illinois University, DeKalb, Illinois, 2000.
65. J. L. Howland, The sign matrix and the separation of matrix eigenvalues, *Lin. Alg. Appl.*, vol. 49, pp. 221-232, 1983.
66. V. Ionescu, C. Oara, and M. Weiss, *Generalized Riccati Theory and Robust Control*, John Wiley, New York, 1999.

67. I. M. Jaimoukha and E. M. Kasenally, Krylov subspace methods for solving large Lyapunov equations, *SIAM J. Numer. Anal.* vol. 31, 227-251, 1994.
68. M. Jamshidi, An overview on the solutions of the algebraic Riccati equation and related problems, *Large-Scale Systems*, vol. 1, pp. 167-192, 1980.
69. C.S. Kenney and G. Hewer, The sensitivity of the algebraic and differential Riccati equations, *SIAM J. Contr. Optimiz.*, vol. 28, pp. 50-69, 1990.
70. C.S. Kenney and A.J. Laub, On scaling Newton's method for polar decomposition and the matrix sign function, *Proc. 1990 Amer. Control Conf.*, pp. 2560-2564, 1990, and *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 688-706, 1992.
71. C.S. Kenney and A.J. Laub, The matrix sign function, *IEEE Trans. Automat. Control*, vol. 40, pp. 1330-1348, 1995.
72. 74C.S. Kenney, A.J. Laub, and M. Wette, A stability-enhancing scaling procedure for Schur-Riccati solvers, *Sys. Contr. Lett.*, vol. 12, pp. 241-250, 1989.
73. C.S. Kenney, A.J. Laub, and M. Wette, Error bounds for Newton refinement of solutions to algebraic Riccati equations, *Math. Control, Signals, and Systems*, vol. 3, pp. 211-224, 1990.
74. H. Kimura, *Chain-scattering Approach to H^∞ -control*, Birkhäuser, Boston, 1996.
75. D. L. Kleinman, On an iterative technique for Riccati equation computations, *IEEE Trans. Automat. Control*, AC-13, pp. 114-115, 1968.
76. D.L. Kleinman, Stabilizing a discrete, constant linear system with application to iterative methods for solving the Riccati equation, *IEEE Trans. Automat. Control*, vol. AC-19, pp. 252-254, 1974.
77. M.M. Konstantinov, P. Petkov, and N.D. Christov, Perturbation analysis of matrix quadratic equations, *SIAM J. Sci. Stat. Comput.*, vol. 11, pp. 1159-1163, 1990.
78. M.M. Konstantinov, P. Petkov, and N.D. Christov, Perturbation analysis of the discrete Riccati equation, *Kybernetika*, vol. 29, pp. 18-29, 1993.
79. M.M. Konstantinov, P. Petkov, D.W. Gu, and I. Postlethwaite, *Perturbation Techniques for Linear Control Problems*, Report 95-7, Control Systems Research, Department of Engineering, Leicester University, UK, 1995.
80. V. Kučera, A contribution to matrix quadratic equations, *IEEE Trans. Automat. Control*, vol. 17, pp. 344-347, 1972.
81. V. Kučera, *Discrete linear control*, John Wiley & Sons, New York, 1979.

82. H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.
83. P. Lancaster and L. Rodman, Existence and uniqueness theorems for algebraic Riccati equations, *Int. J. Control.*, vol. 32, pp. 285-309, 1980.
84. P. Lancaster and L. Rodman, *The Algebraic Riccati Equations*, Oxford University Press, Oxford, 1995.
85. A.J. Laub, A Schur method for solving algebraic Riccati equations, *IEEE Trans. Automat. Control*, AC-24, pp. 913-921, 1979.
86. A.J. Laub, *Invariant subspace methods for the numerical solution of Riccati equations*, in *The Riccati Equation*, pp. 163-196, 1991. (S. Bittanti, et al., Editors).
87. W.-W. Lin, A new method for computing the closed-loop eigenvalues of a discrete-time algebraic Riccati equation, *Lin. Alg. Appl.*, vol. 6, pp. 157-180, 1987.
88. W.-W. Lin, and T.-C. Ho, *On Schur type decompositions of Hamiltonian and Symplectic Pencils*, Tech. Report, Institute of Applied Mathematics, National Tsing Hua University Taiwan, 1990.
89. L. Lu, and W.-W. Lin, An iterative algorithm for the solution of the discrete-time algebraic Riccati equation, *Lin. Alg. Appl.*, vol. 188/189, pp. 465-488, 1993.
90. A. McFarlane, An eigenvector solution of the optimal linear regulator problem, *J. Electronics Control*, vol. 14, pp. 643-654, 1963.
91. V.L. Mehrmann, *The Autonomous Linear Quadratic Control Problem*. Lecture Notes in Control and Information Sciences, vol. 163, Springer-Verlag, Berlin, 1991.
92. V. Mehrmann, A symplectic orthogonal method for single-input single-output discrete-time optimal linear quadratic control problems, *SIAM J. Matrix Anal. Appl.*, pp. 221-248, 1988.
93. C. Paige and C. Van Loan, A Schur decomposition for Hamiltonian matrices, *Lin. Alg. Appl.*, pp. 11-32, 1981.
94. P. Pandey, On scaling an algebraic Riccati equation, *Proc. Amer. Control Conference*, pp. 1583-1587, 1993.
95. T. Pappas, A.J. Laub, and N. Sandell, On the numerical solution of the discrete-time algebraic Riccati equation, *IEEE Trans. Automat. Control*, AC-25, pp. 631-641, 1980.
96. R. V. Patel, On computing the eigenvalues of a symplectic pencil, *Lin. Alg. Appl.* vol. 188/189, pp. 591-611, 1993.

97. R. V. Patel, Z. Lin, and P. Misra. Computation of stable invariant subspaces of Hamiltonian matrices, *SIAM J. Matrix Anal. Appl.*, vol. 15, pp. 284-298, 1994.
98. I.R. Petersen, Disturbance attenuation and H^∞ -optimization: A design method based on the algebraic Riccati equations, *IEEE Trans. Autom. Control*, vol. 32, pp. 427-429, 1987.
99. P. Petkov, N.D. Christov and M.M. Konstantinov, Numerical properties of the generalized Schur approach for solving the discrete matrix Riccati equation, *Proc. 18th Spring Conference of the Union of Bulgarian Mathematicians*, Albena, pp. 452-457, 1989.
100. P. Petkov, N.D. Christov and M.M. Konstantinov, On the numerical properties of the Schur approach for solving the matrix Riccati equation, *Syst. Contr. Lett.*, vol. 9, pp. 197-201, 1987.
101. P. Petkov, N.D. Christov and M.M. Konstantinov, *Computational Methods for Linear Control Systems*, Prentice Hall, London, 1991.
102. P. Petkov, M.M. Konstantinov, D. Gu and I. Postlethwaite, *Solving continuous-time matrix algebraic Riccati equations with condition and accuracy estimates*, Tech. Report 94-64, Control Systems Research, Department of Engineering, Leicester University, Leicester LE1 7RH, UK, Dec. 1994.
103. P. Petkov, M.M. Konstantinov, and V. Mehrmann, DGRSVX and DMSRIC: *Fortran 77 subroutines for solving continuous-time matrix algebraic Riccati equations with condition and accuracy estimates*. Technical Report SFB393/98-116, Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz, FRG, 1998.
104. J.E. Potter, Matrix quadratic solutions, *SIAM J. Appl. Math.*, vol. 14, pp. 496-501, 1966.
105. E. Quintana and V. Hernández, *Algoritmos por bloques y paralelos para resolver ecuaciones matriciales de Riccati mediante el método de Newton*, Tech. Report DSIC-II/6/96, Dpto. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain, 1996a.
106. E. Quintana and V. Hernández, *Algoritmos por bloques y paralelos para resolver ecuaciones matriciales de Riccati mediante el método de Schur*, Tech. Report DSIC-II/7/96, Dpto. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain, 1996b.
107. E. Quintana and V. Hernández, *Algoritmos por bloques y paralelos para resolver ecuaciones matriciales de Riccati mediante la división espectral*, Tech. Report DSIC-II/6/96, Dpto de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain, 1996c.

108. J. Roberts, Linear model reduction and solution of the algebraic Riccati equation by use of the sign function, *Int. J. Control.*, vol. 32, pp. 677-687, 1980 (reprint of a technical report from Cambridge University in 1971).
109. N. Sandell, On Newton's method for Riccati equation solution, *IEEE Trans. Auto. Control*, vol. AC-19, pp. 254-255, 1974.
110. V. Sima, *Algorithms for Linear-Quadratic Optimization*, Marcel Dekker, New York, 1996.
111. V. Sima, An efficient Schur method to solve the stabilizing problem, *IEEE Trans. Automat. Control*, AC-26, pp. 724-725, 1981.
112. V. Sima, P. Petkov and S. Van Huffel, Efficient and reliable algorithms for condition estimation of Lyapunov and Riccati equations, *Proc. Mathematical Theory of Networks and Systems* (MTNS - 2000), 2000.
113. M.A. Singer and S.J. Hammarling, The Algebraic Riccati Equation, *National Physical Laboratory Report*, DITC 23/83, January, 1983.
114. G.W. Stewart, Algorithm 506-HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix, *ACM Trans. Math. Software*, vol. 2, pp. 275-280, 1976.
115. J.-G. Sun, Residual bounds of approximate solutions of the algebraic Riccati equations, *Numer. Math.*, vol. 76, pp. 249-263, 1997a.
116. J.-G. Sun, Backward error for the discrete-time algebraic Riccati equation, *Lin. Alg. Appl.*, vol. 25, pp. 183-208, 1997b.
117. J.-G. Sun, Perturbation analysis of the matrix sign function, *Lin. Alg. Appl.*, vol. 250, pp. 177-206, 1997c.
118. J.-G. Sun, Perturbation theory for algebraic Riccati equation. *SIAM J. Matrix Anal. Appl.*, vol. 19, no. 1, pp. 39-65, 1998.
119. P. Van Dooren, A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Sci. Stat. Comput.*, vol. 2, pp. 121-135, 1981.
120. P. Van Dooren, Algorithm 590-DSUBSP and EXCHQZ: Fortran subroutines for computing deflating subspaces with specified spectrum, *ACM Trans. Math. Software*, vol. 8, pp. 376-382, 1982.
121. C. F. Van Loan, A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix, *Lin. Alg. Appl.*, vol. 16, pp. 233-251, 1984.
122. J.C. Willems, Least squares stationary optimal control and the algebraic Riccati equation. *IEEE Trans. Autom. Control*, AC-16, pp. 621-634, 1971.

123. H.K. Wimmer, The algebraic Riccati equation: Conditions for the existence and uniqueness of solutions. *Lin. Alg. Appl.*, vol. 58, pp. 441-452, 1984.
124. H.K. Wimmer, Existence of positive-definite and semidefinite solutions of discrete-time algebraic Riccati equations. *Int. J. Control.*, vol. 59, pp. 463-471, 1994.
125. S.-F. Xu, Sensitivity analysis of the algebraic Riccati equations, *Numer. Math.*, vol. 75, pp. 121-134, 1996.
126. G. Zames, Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *IEEE Trans. Automat. Control*, AC-26, pp. 301-320, 1981.
127. L.Q. Zhang, J. Lam, and Q.L. Zhang, Lyapunov and Riccati equations of discrete-time descriptor systems, *IEEE Trans. Automat. Control*, vol. 44, no. 1, pp. 2134-2139, 1999.
128. K. Zhou, K. Glover, B. Bodenheimer, and J. Doyle, Mixed H_2 and \mathcal{H}_∞ performance objectives I: Robust performance analysis; *IEEE Trans. Automat. Control*, vol. 39, pp. 1564-1574, 1994.
129. K. Zhou and P. Khargonekar, An algebraic Riccati equation approach to H_∞ optimization, *Syst. Contr. Lett.*, vol. 11, pp. 317-319, 1988.
130. 132 K. Zhou, J. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice Hall, Upper Saddle River, NJ, 1996.

ALGEBRAIC RICCATI EQUATIONS

Contents

13.1 Introduction	585
13.2 The Existence and Uniqueness of the Stabilizing Solution of the CARE	587
13.3 The Existence and Uniqueness of the Stabilizing Solution of the DARE	595
13.4 Conditioning of the Riccati Equations	596
13.4.1 Conditioning of the CARE	596
13.4.2 Conditioning of the DARE	601
13.5 Computational Methods for Riccati Equations	604
13.5.1 The Invariant Subspace Methods	605
13.5.2 The Deflating Subspace Methods	614
13.5.3 The Matrix Sign Function Methods	624
13.5.4 Newton's Methods	631
13.6 The Schur and Inverse-Free Generalized Schur Methods for the Descriptor Riccati Equations	645

13.6.1 The Generalized Schur Method for the DCARE	645
13.6.2 The Inverse-Free Generalized Schur Method for the DCARE	646
13.6.3 The Inverse-Free Generalized Schur Method for the DDARE	646
13.7 Conclusions and Table of Comparisons	647
13.8 Some Selected Software	649
13.8.1 MATLAB CONTROL SYSTEM TOOLBOX	649
13.8.2 MATCONTROL	649
13.8.3 CSP-ANM	649
13.8.4 SLICOT	650
13.8.5 MATRIX _X	650
13.9 Summary and Review	650
13.10 Chapter Notes and Further Reading	654
13.11 Introduction	671
13.12 The Existence and Uniqueness of the Stabilizing Solution of the CARE	673
13.13 The Existence and Uniqueness of the Stabilizing Solution of the DARE	681
13.14 Conditioning of the Riccati Equations	682
13.14.1 Conditioning of the CARE	682
13.14.2 Conditioning of the DARE	687
13.15 Computational Methods for Riccati Equations	690
13.15.1 The Invariant Subspace Methods	691
13.15.2 The Deflating Subspace Methods	700
13.15.3 The Matrix Sign Function Methods	710
13.15.4 Newton's Methods	717
13.16 The Schur and Inverse-Free Generalized Schur Methods for the Descriptor Riccati Equations	731
13.16.1 The Generalized Schur Method for the DCARE	731
13.16.2 The Inverse-Free Generalized Schur Method for the DCARE	732
13.16.3 The Inverse-Free Generalized Schur Method for the DDARE	732
13.17 Conclusions and Table of Comparisons	733
13.18 Some Selected Software	735
13.18.1 MATLAB CONTROL SYSTEM TOOLBOX	735
13.18.2 MATCONTROL	735
13.18.3 CSP-ANM	735
13.18.4 SLICOT	736
13.18.5 MATRIX _X	736
13.19 Summary and Review	736
13.20 Chapter Notes and Further Reading	740

Topics Covered

- Results on Existence and Uniqueness of Solutions of the CARE and DARE
- Perturbation Analyses and Condition Numbers
- The Schur Methods, Newton's Methods, and the Matrix Sign Function Methods
- Convergence Results for Newton's Methods
- The Generalized Eigenvector and the Generalized Schur Methods
- Inverse Free Generalized Schur Methods
- The Schur and Inverse-Free Schur Methods for the Descriptor Riccati Equations
- Comparative Study and Recommendations

13.11 Introduction

This chapter is devoted to the study of numerical solutions of the continuous-time algebraic Riccati equation (CARE):

$$XA + A^T X + Q - XBR^{-1}B^T X = 0 \quad (13.11.1)$$

and of its discrete counterpart (DARE)

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0. \quad (13.11.2)$$

The equation (13.1.1) is very often written in the following compact form:

$$XA + A^T X + Q - XSX = 0 \quad (13.11.3)$$

where

$$S = BR^{-1}B^T. \quad (13.11.4)$$

Similarly, in analogy with the form of the CARE in (13.1.3), the equation (13.1.2) can also be written in the compact form

$$A^T X(I + SX)^{-1} A - X + Q = 0, \quad (13.11.5)$$

where S is again given by (13.1.4).

Throughout this chapter, we assume that $R = R^T > 0$ and $Q = Q^T \geq 0$.

These equations have long been subject of research in mathematics, physics and engineering. They play major roles in many design problems in control and filter theory. As we have seen in Chapter 10, historically, AREs started as an important tool in the solution of *Linear Quadratic Optimization problems*. In recent years, they became a subject of intensive study, both from theoretical and computational viewpoints, because of their important roles in **state-space solutions of H_∞ and robust control problems**. For a brief history of the importance, applications, and historical developments of the algebraic Riccati equations, see Bittanti, et al (1991).

The following computational methods for the CARE and DARE are widely known in the literature and most of them are discussed in **Section 13.5** of this chapter.

- (i) **The Eigenvector Methods** (McFarlane (1963), Potter (1966)).
- (ii) **The Schur-Methods and the Structure-Preserving Schur Methods** (Laub (1979), Byers (1983, 1986a, 1990), Mehrmann (1988), Bunse-Gerstner and Mehrmann (1986), Benner, Mehrmann and Xu (1997)).
- (iii) **The Generalized Eigenvector and the Generalized Schur Methods** (Van Dooren (1981), Arnold and Laub (1984), Pappas, Laub and Sandell (1980), and Mehrmann (1991)).
- (iv) **The Matrix Sign Function Methods** (Roberts (1971, 1980), Bierman (1984), Byers (1987), Denman and Beavers (1976), Kenney and Laub (1995), Gardiner and Laub (1986)).
- (v) **Newton's Methods** (Kleinman (1968), Hewer (1971), Benner and Byers (1994, 1995, 1998), Guo and Lancaster (1998), Guo (1998)).

The eigenvector methods are well-known to have numerical difficulties in case the Hamiltonian matrix associated with the CARE or the symplectic matrix associated with the DARE has some multiple or near-multiple eigenvalues (the corresponding eigenvectors will be ill-conditioned). In these cases, the Schur methods, based on the real Schur decompositions of the Hamiltonian matrix for the CARE and of the symplectic matrix for the DARE, should be preferred over the eigenvector methods. The Schur method is widely used in practice for the CARE. Unfortunately, it cannot be applied to the DARE when A is singular. Indeed, even if A is theoretically nonsingular but is computationally close to a singular matrix, the Schur method for the DARE should be avoided. An alternative for the DARE then is to use the generalized Schur method which is based on the Schur decomposition of a matrix pencil and does not involve computation of the inverse of A . Having said this, it should be noted that the Schur methods and the generalized Schur methods require explicit computation of the inverse of the matrix R both for the CARE and the DARE. So, when R is close to a singular matrix, the methods of choice are the inverse-free generalized Schur methods.

Newton's methods are iterative in nature and are usually used as iterative refinement techniques for solutions obtained by the Schur methods or the matrix sign function methods. **A table of**

comparison of the different methods and recommendation based on this comparison are provided at the end of this chapter.

Sections 13.2 and 13.3 deal, respectively, with the results on the **existence** and **uniqueness of the stabilizing solutions** of the CARE and the DARE. The **condition numbers and bounds of the condition numbers** of the CARE and DARE are identified in Section 13.4.

13.12 The Existence and Uniqueness of the Stabilizing Solution of the CARE

The goal of this section is to derive conditions under which the CARE admits a unique symmetric positive semidefinite stabilizing solution.

For this we first need to develop an important relationship between the CARE and the associated Hamiltonian matrix and some spectral properties of this matrix.

Recall from Chapter 10 that associated with the CARE is the $2n \times 2n$ Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}. \quad (13.12.1)$$

The Hamiltonian matrix H has the following interesting spectral property.

Theorem 13.12.1 *For each eigenvalue λ of H , $-\bar{\lambda}$ is also an eigenvalue of H (with the same geometric and algebraic multiplicity as λ).*

Proof: Define the $2n \times 2n$ matrix

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}, \quad (13.12.2)$$

where I is the $n \times n$ identity matrix. Then it is easy to see that $J^{-1}HJ = -JHJ = -H^T$, which shows that H and $-H^T$ are similar. Hence λ is also an eigenvalue of $-H^T$. Since the eigenvalues of $-H^T$ are the negatives of the eigenvalues of H , and the complex eigenvalues occur in conjugate pairs, the theorem is proved. ■

The following theorems show that a solution X of the CARE is determined by the associated Hamiltonian matrix.

Theorem 13.12.2 *A matrix X is a solution of the CARE if and only if the columns of $\begin{pmatrix} I \\ X \end{pmatrix}$ span an n -dimensional invariant subspace of the Hamiltonian matrix H defined by (13.2.1).*

Proof: We first prove that if the columns of $\begin{pmatrix} I \\ X \end{pmatrix}$ span an n -dimensional invariant subspace of H , then X is a solution of the CARE.

So, assume there exists an $n \times n$ matrix L such that

$$H \begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} I \\ X \end{pmatrix} L. \quad (13.12.3)$$

Multiplying both sides of (13.2.3) by J^{-1} , where J is defined by (13.2.2), we have

$$J^{-1} H \begin{pmatrix} I \\ X \end{pmatrix} = J^{-1} \begin{pmatrix} I \\ X \end{pmatrix} L \quad (13.12.4)$$

Noting that $J^{-1} = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}$, we obtain from (13.2.4)

$$\begin{pmatrix} Q & A^T \\ A & -S \end{pmatrix} \begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} -X \\ I \end{pmatrix} L \quad (13.12.5)$$

Premultiplying both sides of (13.2.5) by $[I, X]$, we get

$$XA + A^T X + Q - XSX = 0,$$

showing that X satisfies the CARE.

To prove the converse, we note that if X is a solution of the CARE, then

$$H \begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} A - SX \\ -Q - A^T X \end{pmatrix} = \begin{pmatrix} A - SX \\ X(A - SX) \end{pmatrix} = \begin{pmatrix} I \\ X \end{pmatrix} (A - SX), \quad (13.12.6)$$

that is, the columns of $\begin{pmatrix} I \\ X \end{pmatrix}$ span an invariant subspace of H .

Corollary 13.12.1 *If the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ span an n -dimensional invariant subspace of the*

Hamiltonian matrix H associated with the CARE and X_1 is invertible, then $X = X_2 X_1^{-1}$ is a solution of the CARE.

Proof:

$$\begin{aligned} \text{The span of the columns of } \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} &= \text{the span of the columns of } \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} X_1^{-1} \\ &= \text{the span of the columns of } \begin{pmatrix} I \\ X_2 X_1^{-1} \end{pmatrix}. \end{aligned}$$

Therefore, by Theorem 13.2.2, we see that $X = X_2 X_1^{-1}$ is a solution of the CARE. ■

The next theorem shows how the eigenvalues of the Hamiltonian matrix H are related to those of the optimal closed-loop matrix.

Theorem 13.12.3 *Let X be a symmetric solution of the CARE. Then the eigenvalues of the Hamiltonian matrix H are the eigenvalues of $A - BK$ together with those of $-(A - BK)^T$, where $K = R^{-1}B^TX$.*

Proof: Define $T = \begin{pmatrix} I & 0 \\ X & I \end{pmatrix}$, where I and X are $n \times n$.

Then

$$\begin{aligned} T^{-1}HT &= \begin{pmatrix} I & 0 \\ -X & I \end{pmatrix} \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} \begin{pmatrix} I & 0 \\ X & I \end{pmatrix} \\ &= \begin{pmatrix} A - SX & -S \\ -(A^TX + XA + Q - XSX) & -(A - SX)^T \end{pmatrix} \\ &= \begin{pmatrix} A - SX & -S \\ 0 & -(A - SX)^T \end{pmatrix} \end{aligned} \quad (13.12.7)$$

Thus the eigenvalues of H are the eigenvalues of $A - SX$ together with those of $-(A - SX)^T$. The result now follows by noting that

$$A - SX = A - BR^{-1}B^TX = A - BK.$$

(Recall that $S = BR^{-1}B^T$). ■

Symmetric Positive Semidefinite Stabilizing Solutions of the CARE

As we have seen in Chapter 10 most applications require a symmetric positive semidefinite stabilizing solution of the associated Riccati equation. Thus, though there exist many solutions of the Riccati equations, we will be developing methods only for the symmetric positive semidefinite stabilizing solutions in this chapter.

We remind the readers that a symmetric solution X of (13.1.1) is a **stabilizing solution** if $A - BK = A - BR^{-1}R^TX = A - SX$ is stable.

For convenience of later use, we now state and prove a necessary and sufficient condition for the existence of such a solution. Proof of Theorem 13.2.4 below has been taken from Kimura (1997).

Theorem 13.12.4 (Existence and Uniqueness of the Stabilizing Solution)

Assume that $R > 0$ and $Q \geq 0, Q \neq 0$.

Then the following conditions are equivalent:

1. The continuous-time algebraic Riccati equation

$$XA + A^TX - XBR^{-1}B^TX + Q = 0, \quad (13.12.8)$$

has a unique symmetric positive semidefinite stabilizing solution X .

2. (A, B) is stabilizable and the associated Hamiltonian matrix H has no pure imaginary eigenvalues.

Proof of Necessity

First suppose that X is a stabilizing solution of the CARE. We then show that H does not have an imaginary eigenvalue.

Since X is a stabilizing solution, $A - SX$ is stable. From Theorem 13.2.3, we then have that n eigenvalues of H are stable and the other n have positive real parts. Thus H does not have a pure imaginary eigenvalue.

Proof of sufficiency

Next assume that H given in (13.2.1), with $S = BR^{-1}B^T$, has no eigenvalues on the imaginary axis. We shall then show that under the assumption of the stabilizability of (A, B) , there exists a unique stabilizing solution of the CARE.

The proof will be divided in several parts.

First of all we note that the stabilizability of (A, B) implies the stabilizability of (A, S) .

Since H has no pure imaginary eigenvalues, there are n stable eigenvalues of H (by Theorem 13.2.1).

Then

$$H \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} E \quad (13.12.9)$$

where E is a stable matrix and the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ form the eigenspace of H corresponding to these stable eigenvalues.

A. $X_2^T X_1$ is symmetric.

The relation (13.2.9) can be expressed as

$$AX_1 - SX_2 = X_1 E \quad (13.12.10)$$

and

$$-QX_1 - A^T X_2 = X_2 E. \quad (13.12.11)$$

Multiplying (13.2.10) by X_2^T on the left, we have

$$X_2^T AX_1 - X_2^T SX_2 = X_2^T X_1 E \quad (13.12.12)$$

Now taking the transpose of (13.2.11), we have

$$X_2^T A = -X_1^T Q - E^T X_2^T$$

Multiplying the last equation by X_1 to the right, we get

$$E^T X_2^T X_1 = -X_2^T A X_1 - X_1^T Q X_1 \quad (13.12.13)$$

Using (13.2.12) in (13.2.13) we then have

$$E^T X_2^T X_1 + X_2^T X_1 E = -X_2^T S X_2 - X_1^T Q X_1 \quad (13.12.14)$$

Since S and Q are symmetric, the right-hand side matrix is symmetric, and therefore the left-hand side matrix is also symmetric. This means that

$$E^T X_2^T X_1 + X_2^T X_1 E = X_1^T X_2 E + E^T X_1^T X_2$$

or

$$E^T (X_2^T X_1 - X_1^T X_2) + (X_2^T X_1 - X_1^T X_2) E = 0.$$

Since E is stable, this Lyapunov equation has a unique solution which implies that $X_2^T X_1 - X_1^T X_2 = 0$. That is $X_2^T X_1 = X_1^T X_2$, proving that $X_2^T X_1$ is symmetric.

B. X_1 is invertible.

Suppose that X_1 is not invertible. Then there exists a vector $d \neq 0$ such that

$$X_1 d = 0. \quad (13.12.15)$$

Now multiplying the transpose of (13.2.10) by d^T to the left and by $X_2 d$ to the right we have

$$\begin{aligned} d^T X_2^T S X_2 d &= -d^T E^T X_1^T X_2 d + d^T X_1^T A^T X_2 d \\ &= -d^T E^T X_2^T X_1 d + d^T X_1^T A^T X_2 d = 0, \\ &\text{(because } X_1^T X_2 = X_2^T X_1 \text{ and } X_1 d = 0\text{).} \end{aligned}$$

Again, since $S \geq 0$, we then must have

$$S X_2 d = 0.$$

The equation (13.2.10) therefore yields

$$X_1 E d = 0.$$

As this holds for all $d \in Ker(X_1)$, this means that $Ker(X_1)$ is E -invariant; that is, there exists an eigenvalue μ of E such that

$$E d' = \mu d', \quad X_1 d' = 0, \quad d' \neq 0. \quad (13.12.16)$$

Again, multiplying (13.2.11) by d' and using the relation (13.2.16) we obtain

$$(\mu I + A^T)X_2d' = 0. \quad (13.12.17)$$

Also, from (13.2.10) and (13.2.16), we have

$$SX_2d' = 0. \quad (13.12.18)$$

Since $\text{Re}(\mu) < 0$ and (A, S) is stabilizable, we conclude from (13.2.18) that

$$X_2d' = 0. \quad (13.12.19)$$

Finally, $X_2d' = 0$ and $X_1d' = 0$ imply that $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ does not have the full rank which contradicts (13.2.9).

Thus X_1 is nonsingular.

C. \mathbf{X} is symmetric.

Since X_1 is nonsingular, we have from Corollary 13.2.1 that $X = X_2X_1^{-1}$ is a solution of the CARE and, since $X_2^TX_1$ is symmetric, so is X . This is seen as follows:

$$\begin{aligned} X^T - X &= X_1^{-T}X_2^T - X_2X_1^{-1} \\ &= X_1^{-T}(X_2^TX_1)X_1^{-1} - X_1^{-T}(X_1^TX_2)X_1^{-1} \\ &= X_1^{-T}(X_2^TX_1 - X_1^TX_2)X_1^{-1} = 0. \end{aligned}$$

D. \mathbf{X} is a stabilizing solution.

Multiplying (13.2.10) by X_1^{-1} to the right we obtain

$$A - SX_2X_1^{-1} = X_1EX_1^{-1}.$$

Since E is stable, so is $A - SX_2X_1^{-1} = A - SX$. Thus X is a stabilizing solution.

E. \mathbf{X} is unique.

Let X_1 and X_2 be two stabilizing solutions. Then

$$\begin{aligned} A^TX_1 + X_1A - X_1SX_1 + Q &= 0 \\ A^TX_2 + X_2A - X_2SX_2 + Q &= 0 \end{aligned}$$

Subtracting these two equations, we have

$$A^T(X_1 - X_2) + (X_1 - X_2)A + X_2SX_2 - X_1SX_1 = 0$$

or

$$(A - SX_1)^T(X_1 - X_2) + (X_1 - X_2)(A - SX_2) = 0$$

Since the last equation is a homogeneous Sylvester equation and the coefficient matrices $A - SX_1$ and $A - SX_2$ are both stable, it follows that $X_1 - X_2 = 0$, that is $X_1 = X_2$.

F. X is positive semidefinite.

Since X is symmetric and satisfies (13.2.8) the equation (13.2.8) can be written in the form of the following Lyapunov equation:

$$(A - BK)^T X + X(A - BK) = -Q - XSX,$$

where $K = R^{-1}B^T X$. Furthermore, $A - BK = A - BR^{-1}B^T X = A - SX$ is stable. Thus, X can be expressed in the form (see Chapter 7):

$$X = \int_0^\infty e^{(A-BK)^T t} (Q + XSX) e^{(A-BK)t} dt.$$

Since Q and S are positive semidefinite, it follows that X is positive semidefinite. ■

Theorem 13.12.5 *Let (A, B) be stabilizable and (A, Q) be detectable. Assume that $Q \geq 0$, $S \geq 0$. Then the Hamiltonian matrix*

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}$$

associated with the CARE does not have a pure imaginary eigenvalue.

Proof: The proof is by contradiction.

Suppose that H has a purely imaginary eigenvalue $j\alpha$, α is a nonnegative real number and let $\begin{pmatrix} r \\ s \end{pmatrix}$ be the corresponding eigenvector. Then

$$H \begin{pmatrix} r \\ s \end{pmatrix} = j\alpha \begin{pmatrix} r \\ s \end{pmatrix}, \quad \begin{pmatrix} r \\ s \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \quad (13.12.20)$$

Multiplying both sides of (13.2.20) by (s^*, r^*) to the left, we obtain

$$s^* Ar - r^* Qr - s^* Ss - r^* A^T s = j\alpha(s^* r + r^* s)$$

or

$$(s^* Ar - r^* A^T s) - r^* Qr - s^* Ss = j\alpha(s^* r + r^* s)$$

Considering the real part of this equation we get

$$-r^* Qr - s^* Ss = 0.$$

Since $S \geq 0$ and $Q \geq 0$ we conclude that

$$Ss = 0 \quad (13.12.21)$$

and

$$Qr = 0. \quad (13.12.22)$$

So, from (13.2.20), we have

$$Ar = j\alpha r \quad (13.12.23)$$

and

$$-A^T s = j\alpha s \quad (13.12.24)$$

Thus combining (13.2.23) and (13.2.22), we have $\begin{pmatrix} A - j\alpha I \\ Q \end{pmatrix} r = 0$.

Since (A, Q) is detectable, we have $r = 0$.

Similarly, using (13.2.24) and (13.2.21) one can show that $s = 0$.

This gives us a contradiction that $\begin{pmatrix} r \\ s \end{pmatrix}$ is an eigenvector. Thus H cannot have a pure imaginary eigenvalue.

■

An Expression for the Stabilizing Solution

Combining Theorem 13.2.4, Corollary 13.2.1 and Theorem 13.2.5, we arrive at the following result:

Theorem 13.12.6 (An Expression for the Unique Stabilizing Solution of the CARE)

Suppose that (A, B) is stabilizable and (A, Q) is detectable. Assume that $Q \geq 0$ and $R > 0$. Then there exists a unique positive semidefinite stabilizing solution X of the CARE: $XA + A^T X - X S X + Q = 0$. This solution is given by $X = X_2 X_1^{-1}$, where the columns of the matrix $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ span the invariant subspace of the Hamiltonian matrix (13.2.1) associated with its stable eigenvalues.

■

Remark: The following simple example shows that the detectability of (A, Q) is not necessary for the existence of a symmetric positive semidefinite stabilizing solution of the CARE.

$$A = \begin{pmatrix} -1 & 0 \\ 0 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad Q = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad R = 1.$$

Then (A, B) is stabilizable, but (A, Q) is not detectable. The matrix $X = \begin{pmatrix} 0 & 0 \\ 0 & 4 \end{pmatrix}$ is the stabilizing solution of the CARE and is positive semidefinite.

13.13 The Existence and Uniqueness of the Stabilizing Solution of the DARE

The existence and uniqueness of the stabilizing solution of the DARE can be studied via a symplectic matrix which takes the role of the Hamiltonian matrix of the CARE.

Definition 13.13.1 A matrix M is **symplectic** if

$$J^{-1}M^TJ = J^TM^TJ = M^{-1},$$

where J is defined by (13.2.2).

Assume A is invertible and consider the matrix

$$M = \begin{pmatrix} A + S(A^{-1})^TQ & -S(A^{-1})^T \\ -(A^{-1})^TQ & (A^{-1})^T \end{pmatrix}, \quad (13.13.1)$$

where $S = BR^{-1}B^T$, $Q = Q^T$, and $S = S^T$.

Then, it can be shown [Exercise 3] that

1. M is symplectic.
2. If λ is a nonzero eigenvalue of M , so is $\frac{1}{\lambda}$.

We now state the discrete counterparts of Theorem 13.2.5 and 13.2.6. The proofs can be found in Lancaster and Rodman (1995).

Theorem 13.13.1 Let (A, B) be discrete-stabilizable and let (A, Q) be discrete-detectable. Assume that $Q \geq 0$ and $S \geq 0$. Then the symplectic matrix (13.3.1) has no eigenvalues on the unit circle.

Suppose that the symplectic matrix M has no eigenvalues on the unit circle. Then it must have n eigenvalues inside the unit circle and n outside it. As in the continuous-time case, it then can be shown that if the columns of the matrix $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ form a basis for the invariant subspace associated with the eigenvalues inside the unit circle, then X_1 is nonsingular and $X = X_2X_1^{-1}$ is a unique symmetric positive semidefinite stabilizing solution of the DARE.

Thus, we have the following theorem as the discrete counterpart of Theorem 13.2.6.

Theorem 13.13.2 (An Expression for the Unique Stabilizing Solution of the DARE) Suppose that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Assume that $Q \geq 0, R > 0$. Then the DARE

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0$$

has a unique symmetric positive semidefinite discrete-stabilizing solution X .

Furthermore X is given by $X = X_2X_1^{-1}$, where the columns of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ span the n -dimensional invariant subspace of the symplectic matrix M associated with the eigenvalues inside the unit circle.

13.14 Conditioning of the Riccati Equations

Before we describe the solution methods for Riccati equations, we state some results on the perturbation theory of such equations that will help us identify the *condition numbers* of the equations. These condition numbers, as usual, will help us understand the sensitivity of the solutions of the Riccati equations when the entries of the data matrices are slightly perturbed.

13.14.1 Conditioning of the CARE

Consider first the CARE:

$$A^T X + X A + Q - X S X = 0, \quad (13.14.1)$$

where

$$S = B R^{-1} B^T. \quad (13.14.2)$$

Let ΔA , ΔX , ΔQ , and ΔS be small perturbations in A , X , Q and S , respectively. Suppose that X is the unique stabilizing solution of the CARE and that $X + \Delta X$ is the unique stabilizing solution of the perturbed Riccati equation

$$(A + \Delta A)^T (X + \Delta X) + (X + \Delta X)(A + \Delta A) + (Q + \Delta Q) - (X + \Delta X)(S + \Delta S)(X + \Delta X) = 0. \quad (13.14.3)$$

We are interested in finding an upper bound for the relative error $\frac{\|\Delta X\|}{\|X\|}$.

Several results exist in literature. Byers (1985) and Kenney and Hewer (1990) obtained the first-order perturbation bounds and Chen (1988), Konstantinov, Petkov, and Christov (1990) gave global perturbation bound. Xu (1996) has improved Chen's result and Konstantinov, Petkov, Gu, and Postlethwaite (1995) have sharpened the results of Konstantinov, Petkov, and Christov (1990). The most recent result in this area is due to Sun (1998), who has improved Xu's result. We present below Sun's result and the condition numbers derived on the basis of this result.

Following the notations in Byers (1985), we define three operators:

$$\Omega(Z) = (A - SX)^T Z + Z(A - SX), \quad (13.14.4)$$

$$\Theta(Z) = \Omega^{-1}(Z^T X + X Z) \quad (13.14.5)$$

$$\Pi(Z) = \Omega^{-1}(X ZX). \quad (13.14.6)$$

Note: Since $A_C = A - SX$ is stable, Ω^{-1} exists. In fact, if $\Omega(Z) = W$, then

$$Z = \Omega^{-1}(W) = - \int_0^\infty e^{A_C^T t} W e^{A_C t} dt,$$

and $\|\Omega^{-1}\|_F = \frac{1}{\text{sep}(A_C^T, -A_C)}.$

Define $l = \|\Omega^{-1}\|^{-1}$, $p = \|\Theta\|$, and $q = \|\Pi\|$, where $\|\cdot\|$ is any unitarily invariant norm. Then the following perturbation result due to Sun (1998) holds:

Theorem 13.14.1 (A Perturbation Bound for the Continuous-time Algebraic Riccati Equation)

Let X and $X + \Delta X$ be, respectively, the symmetric positive semidefinite stabilizing solutions of the CARE (13.4.1) and the perturbed CARE (13.4.3).

Then, for sufficiently small $[\Delta Q, \Delta A, \Delta S]$,

$$\frac{\|\Delta X\|}{\|X\|} \lesssim \frac{\|Q\|}{l\|X\|} \cdot \frac{\|\Delta Q\|}{\|Q\|} + p \frac{\|A\|}{\|X\|} \cdot \frac{\|\Delta A\|}{\|A\|} + \frac{q\|S\|}{\|X\|} \cdot \frac{\|\Delta S\|}{\|S\|}. \quad (13.14.7)$$

■

Using the results of Theorem 13.4.1, Sun has defined a set of condition numbers of the CARE. The numbers

$$\kappa_{CARE}^{AB}(Q) = \frac{1}{l}, \quad \kappa_{CARE}^{AB}(A) = p, \quad \text{and} \quad \kappa_{CARE}^{AB}(S) = q$$

are the **absolute condition numbers** with respect to Q, A, S , respectively.

The numbers $\kappa_{CARE}^{REL}(Q) = \frac{\|Q\|}{l\|X\|}$, $\kappa_{CARE}^{REL}(A) = \frac{p\|A\|}{\|X\|}$ and $\kappa_{CARE}^{REL}(S) = \frac{q\|S\|}{\|X\|}$ are then the **relative condition numbers**.

Moreover, the scalar

$$\kappa_{CARE}^{REL}(X) = \frac{1}{\|X\|} \sqrt{\left(\frac{\|Q\|}{l}\right)^2 + (p\|A\|)^2 + (q\|S\|)^2} \quad (13.14.8)$$

can be regarded as the **relative condition number of X** .

Using a local linear estimate, Byers (1985) has obtained an approximate condition number, given by

$$\kappa_{CARE}^B = \frac{1}{\|X\|_F} \left(\frac{\|Q\|_F}{l} + p\|A\|_F + q\|S\|_F \right),$$

in which the operator norm $\|\cdot\|$ for defining l, p , and q is induced by the Frobenius norm $\|\cdot\|_F$. The above is known as **Byers' approximate condition number**. Indeed, taking the Frobenius norm in (13.4.8), it is easy to see that

Theorem 13.14.2

$$\frac{1}{\sqrt{3}} \kappa_{CARE}^B \leq \kappa_{CARE}^{REL}(X) \leq \kappa_{CARE}^B$$

Expressions for l, p , and q

If the operator norm $\|\cdot\|$ for defining l, p , and q is induced by the Frobenius norm $\|\cdot\|_F$, then it can be shown Sun (1998) that

$$l = \|T^{-1}\|_2^{-1}, \quad p = \|T^{-1}(I_n \otimes X + (X^T \otimes I_n)E)\|_2$$

and

$$q = \|T^{-1}(X^T \otimes X)\|_2,$$

where

$$T = I_n \otimes (A - SX)^T + (A - SX)^T \otimes I_n,$$

and E is the vec-permutation matrix:

$$E = \sum_{i,j=1}^n (e_i e_j^T) \otimes (e_j e_i^T).$$

Remarks. A recent paper of Petkov, Konstantinov and Mehrmann (1998) contains results on estimating the quantities l, p and q .

Bounds on Condition Numbers Using Lyapunov Equations

Computing the quantities l, p , and q using the Kronecker products is computationally intensive. On the other hand, Kenney and Hewer (1990) have obtained an upper and a lower bound of κ_{CARE}^B using solutions of certain Lyapunov equations, which are certainly computationally much less demanding than computing Kronecker products. Using these results, ill-conditioning of κ_{CARE} can be more easily detected.

Assume that $A - SX$ is stable and let H_k be the solution to the Lyapunov equation:

$$(A - SX)^T H_k + H_k (A - SX) = -X^k, \quad k = 0, 1, 2. \quad (13.14.9)$$

Furthermore, let's define $H_1^{(1)}$ as follows:

Set $\tilde{Q} = 2X$, and solve the successive Lyapunov equations:

$$(A - SX)^T \tilde{H} + \tilde{H} (A - SX) = \tilde{Q}. \quad (13.14.10)$$

and

$$(A - SX)H + H(A - SX)^T = \tilde{H}. \quad (13.14.11)$$

Let $W = 2XH$ and $H_1^{(1)} = \Theta\left(\frac{W}{\|W\|}\right)$.

Define

$$U = \frac{\|H_0\| \|Q\| + 2 \|H_0\|^{\frac{1}{2}} \|H_2\|^{\frac{1}{2}} \|A\| + \|H_2\| \|S\|}{\|X\|} \quad (13.14.12)$$

$$L = \frac{\|H_0\| \|Q\| + \|H_1^{(1)}\| \|A\| + \|H_2\| \|S\|}{\|X\|} \quad (13.14.13)$$

Then it can be shown that

$$L \leq \kappa_{CARE}^B \leq U \quad (13.14.14)$$

From the relations (13.4.12)-(13.4.14), we see that κ_{CARE}^B will be large (and consequently the CARE will be ill-conditioned) if H_0 , $H_1^{(1)}$, and H_2 have large norms (relative to that of X). Conversely, if the norms of H_0 , $H_1^{(1)}$, and H_2 are not large, then the CARE will be well-conditioned.

If the norms vary widely in the sense that there is a mixture of large and small norms, then there will be **selective sensitivity**. More specifically, the ratios $r_1 = \frac{\|H_0\| \|Q\|}{\|X\|}$,

$r_2 = \frac{\|H_1^{(1)}\| \|A\|}{\|X\|}$, and $r_3 = \frac{\|H_2\| \|S\|}{\|X\|}$ measure, respectively, the sensitivity of X with respect to perturbations in the matrix Q , the matrix A , and the matrix S .

Example 13.14.1 (An Ill-Conditioned CARE)

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0.0010 & 4 & 5 \\ 0 & 7 & 8 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, R = 1$$

$$Q = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 5 & 3 \\ 1 & 3 & 5 \end{pmatrix}.$$

$$X = 10^9 \begin{pmatrix} 0 & 0.0003 & 0.0004 \\ 0.0003 & 4.5689 & 5.3815 \\ 0.0004 & 5.3815 & 6.3387 \end{pmatrix}.$$

The residual norm of the solution X : $\|XA + A^TX - XSX + Q\| = O(10^{-5})$.

$$\begin{aligned} \|H_0\| &= 5.6491 \times 10^8 \\ \|H_1\| &= 1.8085 \times 10^9 \\ \|H_2\| &= 4.8581 \times 10^{18} \end{aligned}$$

U and L are both of order 10^8 .

Thus the Riccati equation is expected to be ill-conditioned with the given data.

Indeed, this is an example of mixed sensitivity. Note that the ratios r_2 and r_3 are large, but r_1 is quite small. Thus, X should be sensitive with respect to perturbation in A and S . This is verified as follows.

Let M_{new} stand for a perturbed version of the matrix M and X_{new} stands for the new solution of the ARE with the perturbed data.

Case 1. Perturbation in A. Let A be perturbed to $A + \Delta A$, where

$$\Delta A = 10^{-8} \begin{pmatrix} 3.169 & 2.668 & 3.044 \\ -1.259 & -0.5211 & -2.364 \\ 2.798 & 3.791 & -3.179 \end{pmatrix}$$

The matrices B, Q , and R remain unperturbed.

Then

$$\text{Relative error in } X : \frac{\| X_{new} - X \|}{\| X \|} = 4.074 \times 10^{-5}$$

$$\text{Relative perturbation in } A : \frac{\| A_{new} - A \|}{\| A \|} = 4.916 \times 10^{-9}.$$

Case 2. Perturbation in B. Let B be perturbed to $B + \Delta B$,

where

$$\Delta B = 10^{-8} \begin{pmatrix} -4.939 \\ 0.7715 \\ -0.9411 \end{pmatrix}$$

A, Q, R remain unperturbed.

$$\text{Relative error in } X : \frac{\| X_{new} - X \|}{\| X \|} = 7.589 \times 10^{-5}$$

$$\text{Relative perturbation in } B : \frac{\| B_{new} - B \|}{\| B \|} = 5.086 \times 10^{-8}$$

Case 3. Perturbation in Q. The matrix Q is perturbed such that the relative error in $Q : \frac{\| Q_{new} - Q \|}{\| Q \|} = 4.048 \times 10^{-9}$. The matrices A, B , and R remain unperturbed.

$$\text{Then the relative error in } X : \frac{\| X_{new} - X \|}{\| X \|} = 4.048 \times 10^{-9}.$$

Note: All the solutions to the CARE in this example were computed using the Schur method (**Algorithm 13.5.1**) followed by Newton's iterative refinement procedure (**Algorithm 13.5.8**). The residual norms of the solutions obtained by the Schur method alone were of order 10^5 . On the other hand, the residual norm of the solution with the Schur method followed by Newton's iterative procedure was, in each case, of order 10^{-5} .

Example 13.14.2 (A Well-Conditioned CARE).

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{and } R = 1.$$

In this case $\| H_0 \| = 0.3247$, $\| H_1 \| = 0.1251$, $\| H_2 \| = 0.0510$ and $U = 3.1095$

The CARE is, therefore, expected to be well-conditioned.

Indeed, if (1, 1) entry of A is perturbed to -0.9999999 , and the other data remain unchanged, then we find

$$\text{Relative Error in } X: \frac{\| X_{new} - X \|}{\| X \|} = 5.6482 \times 10^{-8}$$

Relative perturbation in A : $\frac{\| A_{new} - A \|}{\| A \|} = 3.1097 \times 10^{-8}$, where A_{new} and X_{new} , respectively, denote the perturbed A and the solution of the CARE with the perturbed data.

Conditioning and Accuracy

Suppose that \hat{X} is an approximate stabilizing solution of the CARE

$$XA + A^T X - XSX + Q = 0,$$

where $S = BR^{-1}B^T$ and let $Res(\hat{X}) = \hat{X}A + A^T\hat{X} - \hat{X}S\hat{X} + Q$.

Then the question arises: If $Res(\hat{X})$ is small, does it guarantee that the error in the solution is also small [Exercise 9]. Recall from Chapter 3 that in the case of linear system problem, it has been shown that the smallness of the residual does not guarantee that the error in the solution is small, if the linear system problem is ill-conditioned. Similar result can be proved in the case of the Riccati equations (see Kenney, Laub, and Wette (1990)). **The result basically says that even if the residual is small, the computed solution may be inaccurate, if the CARE is ill-conditioned.** On the other hand, if $Res(\hat{X})$ is small and the CARE is well-conditioned, then the solution is guaranteed to be accurate. Below, we quote a recent result of Sun (1997a) which is an improvement of the result of Kenney, Laub and Wette (1990).

Theorem 13.14.3 (Residual Bound of an Approximate Stabilizing Solution). Let $\hat{X} \geq 0$ approximate the positive semidefinite stabilizing solution X to the CARE. Define the linear operator $T : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times n}$ by

$$T(Z) = (A - S\hat{X})^T Z + Z(A - S\hat{X}), \quad Z = Z^T \in \mathbb{R}^{n \times n}$$

Assume that $4 \| T^{-1} \| \| T^{-1}(Res(\hat{X})) \| \| S \| < 1$ for any unitarily invariant norm $\| \cdot \|$, then

$$\frac{\| \hat{X} - X \|}{\| \hat{X} \|} \leq \frac{2}{1 + \sqrt{1 - 4 \| T^{-1} \| \| T^{-1}Res(\hat{X}) \| \| S \|}} \frac{\| T^{-1}Res(\hat{X}) \|}{\| \hat{X} \|}.$$

13.14.2 Conditioning of the DARE

Consider now the DARE

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$

The condition number of the DARE, denoted by κ_{DARE} , may be obtained by means of the **Frechet derivative** of the DARE (Gudmundsson, Kenney, and Laub (1992)).

$$\text{Define } A_d = A - B(R + B^T X B)^{-1} B^T X A, \quad S = B R^{-1} B^T. \quad (13.14.15)$$

Assume that X is the stabilizing solution of the DARE. Then the condition number of the DARE is given by:

$$\kappa_{DARE} = \frac{\| [Z_1, Z_2, Z_3] \|_2}{\| X \|_F} \quad (13.14.16)$$

where

$$Z_1 = \| A \|_F P^{-1} (I \otimes A_d^T X + (A_d^T X \otimes I) E), \quad (13.14.17)$$

$$Z_2 = - \| S \|_F P^{-1} (A^T X (I + SX)^{-1} \otimes A^T X (I + SX)^{-1}) \quad (13.14.18)$$

and

$$Z_3 = \| Q \|_F P^{-1}. \quad (13.14.19)$$

In the above, E is the vec-permutation matrix:

$$E = \sum_{i,j=1}^n e_i e_j^T \otimes e_j e_i^T, \quad (13.14.20)$$

and P is a matrix representation of the Stein operator

$$\Omega(Z) = Z - A_d^T Z A_d. \quad (13.14.21)$$

Note that, since A_d is discrete-stable, P^{-1} exists.

The condition number (13.4.16) measures the sensitivity of the stabilizing solution X of the DARE with respect to first-order perturbations.

Assume that the bounds for ΔA , ΔS , and ΔQ are **sufficiently small**. Then, using first-order perturbation only, it can be shown [Exercise 7] that the following quantity is an **approximate condition number** of the DARE

$$\frac{2\|A\|_F^2 \frac{\|Q\|_F}{\|X\|_F} + \|A\|_F^2 \|S\|_F \|X\|_F}{sep_d(A_d^T, A_d)}, \quad (13.14.22)$$

where

$$sep_d(A_d^T, A_d) = \min_{X \neq 0} \frac{\| A_d^T X A_d - X \|_F}{\| X \|_F}. \quad (13.14.23)$$

Note: The quantity $sep(A_d^T, A_d)$ can be computed as the minimum singular value of the matrix

$$A_d^T \otimes A_d^T - I_{n^2}.$$

Remark: A perturbation theorem for the DARE, analogous to Theorem 13.4.1 (for the CARE), and the absolute and relative condition numbers using the results of that theorem, can be obtained. For details see Sun (1998).

Also, a recent paper of Sima, Petkov and Van Huffel (2000) contains efficient and reliable condition number estimators both for the CARE and DARE.

Example 13.14.3 (An Ill-Conditioned DARE). Let's take A, B, Q , and R the same as in Example 13.4.1.

$$\text{Let } A_{\text{new}} = \begin{pmatrix} 0.9999999 & 2 & 3 \\ 0.0010 & 4 & 5 \\ 10^{-8} & 7 & 8 \end{pmatrix}.$$

Let B, Q and R remain unchanged.

The solution X of the DARE (computed by MATLAB function **dare**) is

$$X = 10^{10} \begin{pmatrix} 0.0000 & 0.0005 & 0.0005 \\ 0.0005 & 5.4866 & 6.4624 \\ 0.0005 & 6.4624 & 7.6118 \end{pmatrix}$$

The solution X_{new} of the perturbed version of the DARE is

$$X_{\text{new}} = 10^{10} \begin{pmatrix} 0.0000 & 0.0005 & 0.0005 \\ 0.0005 & 5.4868 & 6.4622 \\ 0.0005 & 6.4622 & 7.6116 \end{pmatrix}$$

Relative error in X : $\frac{\|X - X_{\text{new}}\|}{\|X\|} = 2.3535 \times 10^{-5}$, while the perturbations in A , were of order 10^{-9} .

Example 13.14.4 (A Well-conditioned DARE). Let $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3.999 & 6 & 7 \end{pmatrix}$.

Take B, Q , and R the same as in Example 13.4.1

Let

$$A_{\text{new}} = \begin{pmatrix} 0.9990 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 6 & 7 \end{pmatrix},$$

$B_{\text{new}} = B$, $Q_{\text{new}} = Q$, and $R_{\text{new}} = R$.

Then both the relative error in X and the relative perturbation in A are of $O(10^{-4})$. In this case, $\text{sep}(A_d^T, A_d) = 0.0011$.

13.15 Computational Methods for Riccati Equations

The existing computational methods (**listed in the Introduction**) for the algebraic Riccati equations can be broadly classified into four classes:

- (I) The Invariant Subspace Methods
- (II) The Deflating Subspace Methods
- (III) The Matrix Sign-Function Methods
- (IV) Newton's Methods

The eigenvector and the Schur vector methods are examples of the invariant subspace methods. The generalized eigenvector and the generalized Schur vector methods are examples of the deflating subspace methods.

The following methods have been included in our discussions here.

For the CARE:

- **The eigenvector method** (Section 13.5.1)
- **The Schur method** (Algorithm 13.5.1)
- **The Hamiltonian Schur method** (Section 13.5.1)
- **The inverse-free generalized Schur method** (Algorithm 13.5.3)
- **The matrix sign-function method** (Algorithm 13.5.6)
- **Newton's method** (Algorithm 13.5.8)
- **Newton's method with line search** (Algorithm 13.5.9)

For the DARE:

- **The Schur method** (Section 13.5.1)
- **The generalized Schur method** (Algorithm 13.5.2)
- **The inverse-free generalized Schur method** (Algorithm 13.5.4).
- **The matrix sign function method** (Algorithm 13.5.7)
- **Newton's method** (Algorithm 13.5.10)
- **Newton's method with line search** (Algorithm 13.5.11)

13.15.1 The Invariant Subspace Methods

The invariant subspace methods for the CARE and DARE are, respectively, based on Theorems 13.2.6 and 13.3.2.

Theorem 13.2.6 (Theorem 13.3.2) tells us that a unique stabilizing solution of the CARE (DARE) can be computed by constructing a basis for the stable invariant subspace of the associated Hamiltonian matrix H (the symplectic matrix M). Such bases can be constructed using either the eigendecomposition or the real Schur decomposition of the matrices H and M , giving rise, respectively, to the eigenvector and the Schur vector methods. **We remind the readers that we make the assumption $Q = Q^T \geq 0$ and $R = R^T > 0$ throughout the whole section.**

The Eigenvector Method for the CARE

Assume that (A, B) is stabilizable and (A, Q) is detectable.

Then the Hamiltonian matrix H does not have a purely imaginary eigenvalue. Let H be diagonalizable and have the eigendecomposition:

$$V^{-1}HV = \begin{pmatrix} -\bar{\Lambda} & 0 \\ 0 & \Lambda \end{pmatrix},$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ and $\lambda_1, \dots, \lambda_n$ are the n eigenvalues of H with positive real parts. Let V be partitioned conformably:

$$V = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}$$

such that $\begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix}$ is the matrix of eigenvectors corresponding to the stable eigenvalues. Then it is easy to see that

$$H \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} (-\bar{\Lambda}).$$

From Theorem 13.2.6, we then have that $X = V_{21}V_{11}^{-1}$ is a unique positive semidefinite stabilizing solution.

Remarks. The eigenvector method, in general, cannot be recommended for practical use. The method becomes highly unstable if the Hamiltonian matrix H is defective or nearly defective; that is, if there are some multiple or near multiple eigenvalues of H . In these cases, the matrix V_{11} will be poorly conditioned, making $X = V_{21}V_{11}^{-1}$ inaccurate; and this might happen even if the CARE itself is not ill-conditioned.

The eigenvector method, in principle, is applicable even when H is not diagonalizable by computing the principal vectors, but again is not recommended in practice.

MATCONTROL Note: The eigenvector method for the CARE has been implemented in MATCONTROL function **riceigc**.

The Eigenvector Method for the DARE

An analogous method for the DARE can be developed by taking the eigendecomposition of the associated symplectic matrix M . However, since forming the matrix M requires computation of A^{-1} , **the eigenvector method for the DARE works only when A is nonsingular.** **But even in this case, the results will be inaccurate if A is ill-conditioned.** Moreover, the method will have the same sort of difficulties as those mentioned above for the CARE. We, thus, skip the description of the eigenvector method for the DARE.

The Schur Vector Method for the CARE

The numerical difficulties of the eigenvector method for the CARE may somehow be reduced or eliminated if the Hamiltonian matrix H is transformed to **an ordered Real Schur form** by using the QR iteration algorithm, rather than using its eigendecomposition.

Let $U^T H U$ be an **ordered Real Schur matrix**:

$$U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix},$$

where the eigenvalues of H with negative real parts have been stacked in T_{11} and those with positive real parts are stacked in T_{22} (Note that because of our assumptions that (A, B) is stabilizable and (A, Q) is detectable, H does not have a purely imaginary eigenvalue (**Theorem 13.2.5**)).

Let $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$ be a conformable partitioning of U . Then

$$H \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} T_{11}$$

By Theorem 13.2.6, the matrix $X = U_{21} U_{11}^{-1}$ is then the unique positive semidefinite stabilizing solution of the CARE.

The above discussion leads to the following algorithm, called the *Schur algorithm*, due to Laub (1979).

Algorithm 13.15.1 The Schur Algorithm for the CARE

Inputs: A - The $n \times n$ state matrix
 B - The $n \times m$ input ($m \leq n$) matrix
 Q - The $n \times n$ state weighting matrix
 R - The $m \times m$ control weighting matrix

Output: X - The unique symmetric positive semidefinite stabilizing solution of the CARE.

Assumptions:

1. (A, B) is stabilizable and (A, Q) is detectable.
2. $Q \geq 0, R > 0$.

Step 1. Form the Hamiltonian matrix

$$H = \begin{pmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{pmatrix}.$$

Step 2. Transform H to the **ordered real Schur form**:

$$U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix},$$

where the n eigenvalues of H with negative real parts are contained in T_{11} .

Step 3. Partition U conformably:

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}.$$

Step 4. Compute the solution X by solving the linear systems:

$$XU_{11} = U_{21}.$$

■

Software for the ordered real Schur form:

The ordered real Schur form of H can be obtained by transforming H first to the real Schur form (RSF) by orthogonal similarity, followed by another orthogonal similarity applied to the RSF to achieve the desired ordering of the eigenvalues (**See Chapter 4**).

There exists an efficient algorithm and an associated software developed by Stewart (1976) for this purpose: Algorithm 506 of the Association for Computing Machinery Trans. Math Software (1976), 275-80. See also the LAPACK routine STRSEN.

The MATLAB program **orderschur** from MATCONTROL can also be used for this purpose.

Flop-count. The Schur-method is based on reduction to real Schur form, which is done by QR iterations algorithm; so, an exact flop-count cannot be given. However, assuming that the average number of iterations per eigenvalue is 2, about $234n^3$ flops will be necessary to execute the algorithm, which can be seen as follows:

- 1 Reduction to the Real-Schur Form (including the formation of the matrix U): About $26(2n)^3 = 208n^3$.
2. Solution of $XU_{11} = U_{21}$: $\frac{8}{3}n^3$
3. Ordering the real Schur form: At most $24n^3$

Total: (About) $234n^3$ (These counts are very approximate).

Example 13.15.1 Consider solving the CARE with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad Q = I_{3 \times 3}, \quad S = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

$$B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad R = 1.$$

Step 1. Form the Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} = \left(\begin{array}{ccc|ccc} -1 & 1 & 1 & -1 & -1 & -1 \\ 0 & -2 & 0 & -1 & -1 & -1 \\ 0 & 0 & -3 & -1 & -1 & -1 \\ \hline -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 & 3 \end{array} \right)$$

Step 2. Transform the Hamiltonian matrix to the ordered real Schur form

$$U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

$$= \left(\begin{array}{ccc|ccc} -2.9940 & -0.0216 & 1.3275 & & & * \\ 0 & -2.1867 & 0.7312 & & & \\ 0 & -0.2573 & -1.9055 & & & \\ \hline & 0 & & 2.9940 & 1.3285 & 0.2134 \\ & & & 0 & 1.9623 & 0.2434 \\ & & & 0 & -0.7207 & 2.1298 \end{array} \right)$$

The eigenvalues of T_{11} are: $-2.9940, -2.0461 + 0.4104j, -2.0461 - 0.4104j$. Thus, all the stable eigenvalues are contained in T_{11} .

Step 3. Extract U_{11} and U_{21} from U :

$$U_{11} = \begin{pmatrix} 0.4417 & 0.3716 & 0.7350 \\ 0.0053 & -0.8829 & 0.3951 \\ -0.8807 & 0.1802 & 0.3986 \end{pmatrix}$$

$$U_{21} = \begin{pmatrix} 0.1106 & 0.0895 & 0.3260 \\ 0.0232 & -0.1992 & 0.1552 \\ -0.1285 & 0.0466 & 0.1199 \end{pmatrix}$$

Step 4. Compute the stabilizing solution

$$X = U_{21}U_{11}^{-1} = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}.$$

The eigenvalues of $A - SX$ are: $-2.0461 + 0.4104j, -2.0461 - 0.4104j, -2.9940$. Thus, $A - SX$ is stable; that is, X is a unique stabilizing solution.

MATCONTROL Note. The Schur method for the CARE (using ordered real Schur form) has been implemented in MATCONTROL function **ricschc**.

Stability Analysis of the Schur Method and Scaling

The round-off properties of the Schur method are quite involved. It can be shown (Petkov, et al (1991)) that the relative error in the computed solution is proportional to $\|U_{11}^{-1}\| / \text{sep}(T_{11}, T_{22})$. This means that the Schur method can be numerically unstable even if the CARE is not ill-conditioned.

However, the difficulty can be overcome by proper scaling (Kenney, Laub and Wette (1989)).

Thus, for all practical purposes, the Schur method, when combined with an appropriate scaling, is numerically stable. For a discussion on scaling procedure, see Kenney, Laub, and Wette (1989), and Benner (1997). See also Pandey (1993).

Benner (1997) has given an extensive discussion on scaling. Based on several existing scaling strategies and considering the practical difficulties with these strategies, he has proposed a mixture of these procedures for scaling the CARE. Benner's strategy is as follows:

Write the CARE

$$XA + A^T X - XSX + Q = 0$$

in the form:

$$X_\rho A_\rho + A_\rho^T X_\rho - X_\rho S_\rho X_\rho + Q = 0$$

where $A_\rho = \rho A$, $A_\rho^T = (\rho A)^T$, $X_\rho = \frac{X}{\rho}$, and $S_\rho = \rho^2 S$, ρ being a positive scalar.

Choose ρ as

$$\rho = \begin{cases} \frac{\|S\|_2}{\|Q\|_2} & \text{if } \|Q\|_2 > \|S\|_2 \\ \frac{\|A\|_2}{\|S\|_2} & \text{if } \|Q\|_2 \leq \|S\|_2 \text{ and } \|Q\|_2 \|S\|_2 < \|A\|_2^2 \\ 1 & \text{otherwise.} \end{cases}$$

For a rationale of choosing ρ this way, see Benner (1997).

Note: Note that the relative condition number of the CARE remains invariant under the above scaling.

The Schur Method for the DARE

The Schur method for the DARE

$$A^T X A - X - A^T X B (R + B^T X B)^{-1} B^T X A + Q = 0$$

is analogous. Form the symplectic matrix

$$M = \begin{pmatrix} A + S(A^{-1})^T Q & -S(A^{-1})^T \\ -(A^{-1})^T Q & (A^{-1})^T \end{pmatrix},$$

where $S = BR^{-1}B^T$.

For a unique stabilizing solution of the DARE, we assume that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Then by Theorem 13.3.1, the symplectic matrix M does not have an eigenvalue on the unit circle. In this case M can be transformed to an ordered real Schur form such that the eigenvalues with moduli less than 1 appear in the first block; that is, an orthogonal matrix U can be constructed such that $U^T M U = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix}$, where each eigenvalue of S_{11} is inside the unit circle. Partition U conformably: $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$. Then by Theorem 13.3.2, $X = U_{21} U_{11}^{-1}$ is a unique positive semidefinite stabilizing solution of the DARE.

Remarks: (1) Since one needs to form A^{-1} explicitly to compute M , the Schur method for the DARE is not applicable if A is singular. Even if A is theoretically nonsingular, **the method is expected to give an inaccurate answer in case A is ill-conditioned with respect to inversion.**

(2) A slightly faster method (Sima (1996), p. 244) forms the matrix M^{-1} and orders the Real Schur form so that the eigenvalues with modulii less than 1 appear in the first block.

MATCONTROL Note. The Schur method for the DARE has been implemented in MATCONTROL function **ricschd**.

The Hamiltonian-Schur Methods for the CARE

The Schur methods for the algebraic Riccati equations are based on orthogonal similarity transformations of the associated Hamiltonian and symplectic matrices to real Schur forms. The rich structures of these matrices are, however, not exploited in these methods. The Hamiltonian and the symplectic matrices are treated just as $2n \times 2n$ general matrices in these methods. **It would be useful if methods could be developed that could take advantage of Hamiltonian and Symplectic structures.** Such structure-preserving methods, besides reflecting physical structures, are often faster.

Theorem 13.5.1 below shows that developments of such structure-preserving methods are possible.

Definition 13.15.1 If a matrix U is both symplectic and unitary, it is called a **symplectic-unitary matrix**. A **symplectic-orthogonal** matrix can similarly be defined.

From the above definition, it follows that an $2n \times 2n$ symplectic-unitary matrix U can be written as

$$U = \begin{pmatrix} U_{11} & U_{12} \\ -U_{12} & U_{11} \end{pmatrix},$$

where U_{11} and U_{12} are $n \times n$. If \hat{U} is $n \times n$ unitary, then

$$U = \begin{pmatrix} \hat{U} & 0_{n \times n} \\ 0_{n \times n} & \hat{U} \end{pmatrix}$$

is symplectic-unitary.

Theorem 13.15.1 (The Hamiltonian-Schur Decomposition (HSD) Theorem) (Paige and Van Loan (1981)). If the real parts of all the eigenvalues of a Hamiltonian matrix H are nonzero, then there exists a symplectic-orthogonal matrix U and a Hamiltonian matrix T such that

$$U^T H U = T = \begin{pmatrix} T_1 & T_2 \\ 0_{n \times n} & -T_1^T \end{pmatrix},$$

where T_1 is $n \times n$ upper triangular, and T_2 is $n \times n$ symmetric. Furthermore, U can be chosen so that the eigenvalues of T_1 have negative real parts.

Definition 13.15.2 The Hamiltonian matrix T in Theorem 13.5.1 is called a **Hamiltonian-Schur matrix** and the decomposition itself is called **Hamiltonian-Schur Decomposition**.

Note. The first n columns of U in the above Hamiltonian-Schur decomposition span the invariant subspace corresponding to the stabilizing solution of the CARE.

Symplectic-Schur Decomposition (SSD)

For a symplectic matrix, we have the following Theorem.

Theorem 13.15.2 (The Symplectic-Schur Decomposition (SSD) Theorem). If M is symplectic and has no eigenvalues on the unit circle, then there exists a symplectic-orthogonal matrix U such that

$$U^T M U = R = \begin{pmatrix} R_1 & R_2 \\ 0_{n \times n} & R_1^{-T} \end{pmatrix}$$

where R_1 is $n \times n$ upper triangular. Moreover, $R_2 R_1$ is symmetric.

Definition 13.15.3 The above decomposition is called a **Symplectic-Schur Decomposition**.

The existence of the Hamiltonian-Schur decomposition and the symplectic-Schur decomposition naturally lead to the following problem: **How to obtain these decompositions in a numerically effective way by exploiting the structures of the Hamiltonian and the symplectic matrices?**

Byers (1983, 1986a) first developed such a structure-preserving method for the Hamiltonian Schur decomposition in the case the matrix Q in the Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix},$$

has rank 1. By permutation, the method is also applicable if $\text{rank}(S) = 1$.

Definition 13.15.4 *A Hamiltonian matrix H has **Hamiltonian-Hessenberg form**, if it has the zero structure of a $2n \times 2n$ upper Hessenberg matrix with the order of the last n rows and columns reversed.*

As in the standard QR iteration algorithm for the real Schur form of a matrix A , Byers' method also comes in two stages:

Stage I. The matrix H is reduced to a **Hamiltonian-Hessenberg matrix** H_H by an orthogonal-symplectic transformation.

Stage II. The Hamiltonian-Hessenberg matrix H_H is further reduced to Hamiltonian-Schur form using Hamiltonian QR iterations.

Of course, once such a reduction is done, this can immediately be used to solve the CARE. For a complete description of the method and details of numerical implementations, see Byers (1986a).

Unfortunately, in spite of several attempts, such a reduction in the general case of a Hamiltonian matrix remained a difficult problem, until the recent paper of Benner, Mehrmann and Xu (1997).

A Hamiltonian-Schur Method for the CARE ($\text{rank } S \geq 1$)

We next outline briefly the Hamiltonian-Schur method of Benner, Mehrmann and Xu (1997) for solving the CARE in the multi-input case. The method also uses symplectic-orthogonal transformations in the reduction to the Hamiltonian Schur form of the matrix H_E defined below.

The method is based on an interesting relationship between the invariant subspaces of the Hamiltonian matrix H and the extended matrix $\begin{pmatrix} 0 & H \\ H & 0 \end{pmatrix}$. It makes use of the **symplectic URV-like decomposition** that was also introduced by the authors (Benner, Mehrmann and Xu (1999)).

Theorem 13.15.3 (Symplectic-URV Decomposition)

Given a $2n \times 2n$ Hamiltonian matrix H , there exist symplectic-orthogonal matrices U_1 and U_2 such that

$$H = U_2 \begin{pmatrix} H_t & H_r \\ 0 & -H_b^T \end{pmatrix} U_1^T,$$

where H_t is an $n \times n$ upper triangular matrix and H_b is an $n \times n$ real Schur matrix.

Furthermore, the positive and negative square roots of the eigenvalues of $H_t H_b$ are the eigenvalues of H . ■

The basis of the Hamiltonian-Schur method is the following result.

Theorem 13.15.4 (Extended Hamiltonian Schur Decomposition Theorem). Suppose that the Hamiltonian matrix

$$H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}$$

has no purely imaginary eigenvalues. Define

$$H_E = \begin{pmatrix} 0 & H \\ H & 0 \end{pmatrix}.$$

Then there exists, an orthogonal matrix U of order $4n$ such that

$$U^T H_E U = T = \begin{pmatrix} T_1 & T_2 \\ 0 & -T_1^T \end{pmatrix}$$

is in Hamiltonian-Schur form and no eigenvalues of T_1 have negative real parts. ■

Remark: Note that the transforming matrix U in Theorem 13.5.4 is not symplectic-orthogonal. But this non-symplectic transformation can be computed without rounding errors!

Solution of the CARE using the Extended Hamiltonian Schur Decomposition

Let the matrix U in Theorem 13.5.4 be partitioned as

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$$

where each U_{ij} is of order $2n \times 2n$. Define the matrix \hat{Y} as

$$\hat{Y} = \frac{\sqrt{2}}{2}(U_{11} - U_{21}).$$

Let Y be an orthogonal basis of $\text{Range}(\hat{Y})$. Then it has been shown (Benner, Mehrmann and Xu (1997)) that

$$\text{Range}(\hat{Y}) = \text{Inv}(H),$$

where $\text{Inv}(H)$ is the invariant subspace associated with the eigenvalues of H with negative real parts.

Furthermore, if

$$\hat{Y} = \begin{pmatrix} \hat{Y}_1 \\ \hat{Y}_2 \end{pmatrix},$$

where \hat{Y}_1 and \hat{Y}_2 are of order $n \times 2n$, then the stabilizing solution X of the CARE is given by

$$X\hat{Y}_1 = -\hat{Y}_2.$$

Note that the above equations represent an overdetermined consistent set of linear equations.

The symplectic-URV decomposition is used to compute the matrix U to achieve the Hamiltonian-Schur matrix T . Note also that it is not necessary to explicitly compute Y , if only the stabilizing solution of the CARE is sought.

The details are rather involved and we refer the readers to the paper of Benner, Mehrmann, and Xu (1997).

Efficiency and stability: The method based on the above discussion is more efficient than the Schur method. It has also been shown that the method computes the Hamiltonian-Schur form of a Hamiltonian matrix close to \tilde{H}_E , where \tilde{H}_E is permutationally similar to H_E , that is, there exists a permutation matrix P such that $PH_EP^T = \tilde{H}_E$.

Remark: Note that the above method is not structure preserving for the Hamiltonian matrix H but it is structure preserving for \tilde{H}_E .

13.15.2 The Deflating Subspace Methods

The deflating subspace methods are generalizations of the invariant subspace methods in the sense that the solutions of the Riccati equations are now computed by finding the bases for the deflating subspaces of certain matrix pencils rather than finding those of the Hamiltonian and the symplectic matrices.

For the CARE, the pencil is $P_{CARE} - \lambda N_{CARE}$, where

$$P_{CARE} = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}, \quad N_{CARE} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}. \quad (13.15.1)$$

For the DARE, the pencil is $P_{DARE} - \lambda N_{DARE}$, where

$$P_{DARE} = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}, \quad N_{DARE} = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}. \quad (13.15.2)$$

Since no inversion of A is required to form the above pencils, **this generalization is significant for the DARE**, because, as we have seen, the eigenvector and the Schur methods cannot be applied to the DARE when A is singular.

As in the case of an invariant subspace method, a basis for a deflating subspace of a pencil can be constructed either by using the generalized eigendecomposition or the generalized Schur decomposition of the pencil. As before, an eigenvector method will have numerical difficulties in case the pencil has a multiple or near-multiple eigenvalue. **We will thus skip the descriptions of the generalized eigenvector methods and describe here only the generalized Schur method for the DARE.** We leave the description of the generalized Schur method for the CARE as an exercise [Exercise 19].

The following results form a mathematical foundation for a deflating subspace method for the DARE. The results are due to Pappas, Laub and Sandell (1980).

Theorem 13.15.5 Suppose that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Then the symplectic pencil $P_{\text{DARE}} - \lambda N_{\text{DARE}}$ does not have any eigenvalue λ with $|\lambda| = 1$.

Proof: The proof is by contradiction.

Let $|\lambda| = 1$ be an eigenvalue of the pencil $P_{\text{DARE}} - \lambda N_{\text{DARE}}$ with the eigenvector $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \neq 0$. Then we can write

$$\begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \lambda \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}.$$

This means that

$$Az_1 = \lambda z_1 + \lambda S z_2 \quad (13.15.3)$$

$$-Qz_1 + z_2 = \lambda A^T z_2. \quad (13.15.4)$$

Premultiplying the first equation by $\bar{\lambda} z_2^*$ and postmultiplying the conjugate transpose of the second by z_1 we have

$$\bar{\lambda} z_2^* Az_1 = |\lambda|^2 z_2^* z_1 + |\lambda|^2 z_2^* S z_2 \quad (13.15.5)$$

and

$$z_2^* z_1 = z_1^* Q z_1 + \bar{\lambda} z_2^* A z_1 \quad (13.15.6)$$

Substituting (13.5.5) into (13.5.6), we obtain

$$z_2^* z_1 = z_1^* Q z_1 + |\lambda|^2 z_2^* z_1 + |\lambda|^2 z_2^* S z_2 \quad (13.15.7)$$

or

$$z_2^* S z_2 + z_1^* Q z_1 = 0 \quad (\text{since } |\lambda|^2 = 1). \quad (13.15.8)$$

Since $S = BR^{-1}B^T$, the equation (13.5.8) can be written as

$$(z_2^* B) R^{-1} (B^T z_2) + z_1^* Q z_1 = 0. \quad (13.15.9)$$

Since R is positive definite, this implies that

$$B^T z_2 = 0 \text{ and } Qz_1 = 0. \quad (13.15.10)$$

Therefore, from (13.5.3) and (13.5.4), we have $Az_1 = \lambda z_1$ and $A^T z_2 = \frac{1}{\lambda} z_2$. (Note that since $|\lambda| = 1, \lambda \neq 0$).

Thus, from (13.5.10) and from the last equation, we have $z_2^* B = 0$ and $z_2^* A = \frac{1}{\lambda} z_2^*$.

This means that for any F , $z_2^*(A - BF) = \frac{1}{\lambda} z_2^*$; that is, $\frac{1}{\lambda}$ is an eigenvalue of $A - BF$ for every F . Since (A, B) is discrete-stabilizable, this means that $z_2 = 0$. Similarly, since (A, Q) is detectable, it can be shown [Exercise 18] that $z_1 = 0$. Therefore, $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$ is a zero vector, which is a contradiction. ■

Theorem 13.5.5 together with the fact that if $\lambda \neq 0$ is an eigenvalue with multiplicity r of the pencil $P_{DARE} - \lambda N_{DARE}$, so is $\frac{1}{\lambda}$ with the same multiplicity, allows us to state the following theorem:

Theorem 13.15.6 *Suppose that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Let $\lambda = 0$ be an eigenvalue of multiplicity r . Then the eigenvalues of the pencil $P_{DARE} - \lambda N_{DARE}$ can be arranged as follows (adopting the convention that the reciprocal of a zero is infinity):*

$$\underbrace{0, \dots, 0}_r; \quad \underbrace{\lambda_{r+1}, \dots, \lambda_n}_{n-r}; \quad \underbrace{\frac{1}{\lambda_n}, \dots, \frac{1}{\lambda_{r+1}}}_{n-r}; \quad \underbrace{\infty, \infty, \dots, \infty}_r.$$

with $0 < |\lambda_i| < 1, i = r+1, \dots, n$. ■

MATCONTROL Note. The generalized eigenvector method for the DARE has been implemented in MATCONTROL function **ricgeigd**.

The Generalized Schur-Vector Method for the DARE

Assume that the generalized Schur form of the pencil $P_{DARE} - \lambda N_{DARE}$ has been ordered such that the generalized eigenvalues of the pencil with moduli less than 1 can be obtained from the first quarters of the matrices; that is, the orthogonal matrices Q' and Z have been computed such that

$$Q'(P_{DARE} - \lambda N_{DARE})Z = P_1 = \begin{pmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{pmatrix}$$

and

$$Q'(P_{DARE} - \lambda N_{DARE})Z = N_1 = \begin{pmatrix} N_{11} & N_{12} \\ 0 & N_{22} \end{pmatrix}$$

and the generalized eigenvalues of the pencil $P_{11} - \lambda N_{11}$ have moduli less than 1 (see below for details of how to do this).

Let $Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$. Then the columns of $\begin{pmatrix} Z_{11} \\ Z_{21} \end{pmatrix}$ form a basis for the discrete stable (deflating) subspace and the matrix $X = Z_{21}Z_{11}^{-1}$ is a unique symmetric positive semidefinite stabilizing solution of the DARE. We leave the details as an exercise [Exercise 19].

Algorithm 13.15.2 The Generalized Schur Algorithm for the DARE

Inputs: A – The $n \times n$ state matrix
 B – The $n \times m$ input matrix
 Q – The $n \times n$ state weighting
 R – The $m \times m$ control weighting matrix

Output: X - The unique $n \times n$ symmetric positive semidefinite stabilizing solution of the DARE: $A^T X A + Q - X - A^T X B (R + B^T X B)^{-1} B^T X A = 0$

Assumptions:

1. (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable.
2. $Q \geq 0, R > 0$.

Step 1. Form $P_{DARE} = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}$, $N_{DARE} = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}$.

Step 2. Transform the pencil $P_{DARE} - \lambda N_{DARE}$ to the **generalized real Schur form** using the QZ algorithm; that is find orthogonal matrices Q_1 and Z_1 such that

$$Q_1 P_{DARE} Z_1 = P_1 = \begin{pmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{pmatrix}$$

and

$$Q_1 N_{DARE} Z_1 = N_1 = \begin{pmatrix} N_{11} & N_{12} \\ 0 & N_{22} \end{pmatrix},$$

where P_1 is quasi-upper triangular and N_1 is upper triangular.

Step 3.

Reorder the above generalized real Schur form by using an orthogonal transformation, so that the pencil $P_{11} - \lambda N_{11}$ has all its eigenvalues with moduli less than 1. That is, find orthogonal matrices Q_2 and Z_2 such that $Q_2 Q_1 P_{DARE} Z_1 Z_2$ is quasi-upper triangular and $Q_2 Q_1 N_{DARE} Z_1 Z_2$ is upper triangular and moreover, the diagonal blocks corresponding to the eigenvalues with modulii less than 1 are in the upper left quarter of these matrices.

Step 4. Form $Z = Z_1 Z_2 = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$.

Step 5. Compute $X = Z_{21} Z_{11}^{-1}$; that is, solve for X : $XZ_{11} = Z_{21}$.

Example 13.15.2 Consider solving the DARE with

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = 1.$$

$$\text{Step 1. } P_{DARE} = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}, \quad N_{DARE} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 2 & 4 \end{pmatrix}.$$

Step 2. The generalized Real Schur form of the pencil $P_{DARE} - \lambda N_{DARE}$ is given by:

$Q_1(P_{DARE} - \lambda N_{DARE})Z_1 = P_1 - \lambda N_1$, where

$$P_1 = \begin{pmatrix} -5.5038 & 0.3093 & 0.7060 & 0.0488 \\ 0 & 1.4308 & 0.1222 & 0.0903 \\ 0 & 0 & 0.2665 & 0.2493 \\ 0 & 0 & 0 & 0.9530 \end{pmatrix}, \quad N_1 = \begin{pmatrix} -0.9912 & -0.3540 & 0.2965 & -0.8012 \\ 0 & -0.2842 & 0.8565 & -0.5442 \\ 0 & 0 & -1.3416 & 0.9885 \\ 0 & 0 & 0 & 5.2920 \end{pmatrix}$$

Step 3. The eigenvalues with modulii less than 1 are :

$$\frac{P_1(3,3)}{N_1(3,3)} = -0.1986 \text{ and } \frac{P_1(4,4)}{N_1(4,4)} = 0.1801.$$

Step 4. The matrix $\begin{pmatrix} Z_{11} \\ Z_{21} \end{pmatrix}$ is given by

$$\begin{pmatrix} Z_{11} \\ Z_{21} \end{pmatrix} = \begin{pmatrix} 0.5518 & -0.1074 \\ -0.3942 & 0.0847 \\ 0.6400 & 0.4499 \\ -0.3614 & 0.8825 \end{pmatrix}$$

Step 5. $X = Z_{21} Z_{11}^{-1} = \begin{pmatrix} 54.9092 & 75.2247 \\ 75.2247 & 106.1970 \end{pmatrix}$ is the stabilizing solution.

Implementational Details

The reduction to the generalized real Schur form can be achieved using the QZ algorithm, as described in Chapter 4.

Unfortunately, however, the eigenvalues might appear in any arbitrary order. Some reordering needs to be done. A systematic way to do this is as follows:

First, check if the last eigenvalue in the upper left quarter has modulus less than 1, if not, move it to the last position in the lower right quarter. Check the next eigenvalue now in the upper left quarter, if it does not have modulus less than 1, move it to the next position in the lower right quarter.

Note that each move is equivalent to finding a pair of orthogonal matrices such that pre- and post-multiplications by these matrices perform the necessary change.

The process can be continued until all the n eigenvalues with moduli greater than 1 have been moved to the lower right quarter and the upper left quarter contains only the eigenvalues with moduli less than 1.

There is also a slightly more efficient algorithm (Sima (1996), pp. 262-264) for ordering the eigenvalues of the pencil $P_{DARE} - \lambda N_{DARE}$.

There exists FORTRAN Routines, developed by Van Dooren (1982) to compute deflating subspaces with specified spectrum. These subroutines are available as Algorithm 590–DSUBSP and EXCHQZ in ACM software library. Also, the LAPACK package (Anderson et al. (1995)) includes the routine STGSEN, which performs a specified reordering of the eigenvalues of the generalized real Schur form.

Numerical stability and scaling. It can be shown (see Petkov et al. (1989)) that the generalized Schur method may yield inaccurate results if the DARE is not properly scaled. For a scaling strategy that can be used to overcome this problem, see Gudmundsson et al. (1992) and Benner (1997).

The Generalized Schur Methods Without Explicit Computation of the Inverse of the Control Weighting Matrix R .

All the methods we have considered so far require the explicit computation of the inverse of the control weighting matrix R . **These methods, therefore, may not yield accurate solutions when R is severely ill-conditioned.**

For example, consider the following example from Arnold and Laub (1984):

$$A = \begin{pmatrix} -0.1 & 0 \\ 0 & -0.02 \end{pmatrix}, B = \begin{pmatrix} 0.1 & 0 \\ 0.001 & 0.01 \end{pmatrix}, Q = \begin{pmatrix} 100 & 1000 \\ 1000 & 10000 \end{pmatrix}, R = \begin{pmatrix} 1 + \epsilon & 1 \\ 1 & 1 \end{pmatrix}.$$

The pair (A, B) is controllable. The matrix R becomes progressively ill-conditioned as $\epsilon \rightarrow 0$. The CARE with the above data was solved by Arnold and Laub, using RICPACK, a software package especially designed for solving Riccati equations. It was shown that the accuracy of the solution deteriorated as R became more and more ill-conditioned. For $\epsilon = 10^{-16}$ the relative accuracy was of 10^{-1} only.

In this case, an **inverse-free generalized Schur method**, that avoids computations of R^{-1} is useful.

The Continuous-Time Case

First, we observe that the Hamiltonian eigenvalue problem $Hx = \lambda x$ associated with the CARE, can be replaced by the eigenvalue problem for the extended $(2n + m) \times (2n + m)$ pencil:

$$P_{CARE}^E - \lambda N_{CARE}^E,$$

where $P_{CARE}^E = \begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix}$, and $N_{CARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix}$.

(Note that this pencil does not involve R^{-1}). The solution of the CARE can now be obtained by constructing a basis of the stable deflating subspace of this pencil. It was further observed by Van Dooren (1981) that this $(2n + m) \times (2n + m)$ can be compressed, using an orthogonal factorization of the matrix $\begin{pmatrix} R \\ B \end{pmatrix}$, into a $2n \times 2n$ pencil, without affecting the deflating subspaces. Thus, if

$$\begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} \begin{pmatrix} R \\ B \end{pmatrix} = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix},$$

then instead of considering the $(2n + m) \times (2n + m)$ pencil $P_{CARE}^E - \lambda N_{CARE}^E$, we consider the $2n \times 2n$ compressed pencil $P_{CARE}^{EC} - \lambda N_{CARE}^{EC}$, where $P_{CARE}^{EC} = \begin{pmatrix} W_{22}A & W_{21}B^T \\ -Q & -A^T \end{pmatrix}$ and $N_{CARE}^{EC} = \begin{pmatrix} W_{22} & 0 \\ 0 & I \end{pmatrix}$.

This leads to the following algorithm:

Algorithm 13.15.3 Inverse-Free Generalized Schur Algorithm for the CARE.

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix ($m \leq n$)

Q - The $n \times n$ state weighting matrix

R - The $m \times m$ control weighting matrix

Output: X - The unique symmetric positive semidefinite stabilizing solution of the CARE

Assumptions: 1) (A, B) is stabilizable, and (A, Q) is detectable.

2) $Q \geq 0, R > 0$.

Step 1. Find the QR factorization of the matrix $\begin{pmatrix} R \\ B \end{pmatrix}$:

$$W \begin{pmatrix} R \\ B \end{pmatrix} = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}.$$

Partition $W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$, where W_{22} is an $n \times n$ matrix.

Step 2. Form P_{CARE}^{EC} and N_{CARE}^{EC} as shown above.

Step 3. Find the ordered generalized Schur form of the pencil $P_{CARE}^{EC} - \lambda N_{CARE}^{EC}$ using the QZ algorithm; that is, find orthogonal matrices Q_1 and Z such that $Q_1(P_{CARE}^{EC} - \lambda N_{CARE}^{EC})Z = \tilde{M} - \lambda \tilde{N}$; where \tilde{M} and \tilde{N} are, respectively, quasi-upper and upper triangular matrices, and the n eigenvalues with negative real parts appear first.

Step 4. Compute $X = Z_{21}Z_{11}^{-1}$ where $Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$. ■

Remark: In his paper, Van Dooren (1981) described the compression technique by using an orthogonal factorization of the matrix $\begin{pmatrix} B \\ 0 \\ R \end{pmatrix}$.

Instead, here we have used (an equivalent) factorization of $\begin{pmatrix} R \\ B \end{pmatrix}$ in the form $\begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}$, so that a standard QR factorization algorithm can be used to achieve this factorization.

Example 13.15.3

$$A = \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = 10^{-10}$$

Step 1.

$$W = \left(\begin{array}{c|cc} -0.0000 & -1.0000 & 0 \\ \hline -1.0000 & 0.0000 & 0 \\ 0 & 0 & 1.0000 \end{array} \right) = \left(\begin{array}{c|c} W_{11} & W_{12} \\ \hline W_{21} & W_{22} \end{array} \right)$$

Step 2.

$$P_{CARE}^{EC} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & -2 & -1 \\ 0 & -1 & 1 & 0 \end{pmatrix}$$

$$N_{CARE}^{EC} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Step 3.

$$Z = \begin{pmatrix} -1.0000 - 0.0028i & 0.0000 + 0.0000i & -0.0000 + 0.0000i & -0.0000 + 0.0000i \\ 0.0000 + 0.0000i & 0.7071 + 0.0025i & -0.7071 + 0.0044i & 0.0000 - 0.0000i \\ -0.0000 - 0.0000i & 0.0000 + 0.0000i & 0.0000 - 0.0000i & 1.0000 - 0.0000i \\ 0.0000 + 0.0000i & 0.7071 + 0.0025i & 0.7071 - 0.0044i & -0.0000 + 0.0000i \end{pmatrix}$$

Step 4.

$$X \text{ (in Long Format)} = \begin{pmatrix} 0.00001000030018 & 0.00000999990018 \\ 0.00000999990018 & 1.00001000029721 \end{pmatrix}.$$

Verify: The residual norm= 7.357×10^{-8}

The Discrete-Time Case

The discrete problem is analogous. Here we consider the $(2n+m) \times (2n+m)$ pencil $P_{DARE}^E - \lambda N_{DARE}^E$, where

$$P_{DARE}^E = \begin{pmatrix} A & 0 & -B \\ -Q & -I & 0 \\ 0 & 0 & R \end{pmatrix},$$

and

$$N_{DARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & A^T & 0 \\ 0 & B^T & 0 \end{pmatrix}.$$

This pencil is then compressed into the $2n \times 2n$ pencil $P_{DARE}^{EC} - \lambda N_{DARE}^{EC}$ where

$$P_{DARE}^{EC} = \begin{pmatrix} W_{22}A & 0 \\ -Q & -I \end{pmatrix},$$

and

$$N_{DARE}^{EC} = \begin{pmatrix} W_{22} & W_{21}B^T \\ 0 & A^T \end{pmatrix},$$

by taking the QR factorization of the matrix $\begin{pmatrix} R \\ -B \end{pmatrix}$:

$$W \begin{pmatrix} R \\ -B \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \text{ where } W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}.$$

This leads to the following algorithm:

Algorithm 13.15.4 Inverse-free Generalized Schur Method for the DARE.

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix ($m \leq n$)

Q - The $n \times n$ state weighting matrix

R - The $m \times m$ control weighting matrix

Output: X - The unique symmetric positive semidefinite stabilizing solution of the DARE.

Assumptions: 1) (A, B) is stabilizable and (A, Q) is detectable.

2) $Q \geq 0, R > 0$.

Step 1. Find the QR factorization of $\begin{pmatrix} R \\ -B \end{pmatrix}$; that is find an orthogonal matrix W such that

$$W \begin{pmatrix} R \\ -B \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}.$$

$$\text{Partition } W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}.$$

$$\text{Step 2. Form } P_{DARE}^{EC} = \begin{pmatrix} W_{22}A & 0 \\ -Q & -I \end{pmatrix}, \quad N_{DARE}^{EC} = \begin{pmatrix} W_{22} & W_{21}B^T \\ 0 & A^T \end{pmatrix}.$$

Step 3. Compute the **ordered generalized Schur form** of the pencil $P_{DARE}^{EC} - \lambda N_{DARE}^{EC}$, using the QZ algorithm followed by some ordering procedure so that the eigenvalues of moduli less than 1 appear in the first quarter; that is, find orthogonal matrices Q_1 and Z such that $Q_1(P_{DARE}^{EC} - \lambda N_{DARE}^{EC})Z = \tilde{P} - \lambda \tilde{N}$ and the n eigenvalues with modulii less than 1 appear first.

$$\text{Step 4. Form } X = Z_{21}Z_{11}^{-1} \text{ where } Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix},$$

Example 13.15.4 Consider solving the DARE with

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}, R = 1.$$

Step 1.

$$W = \left(\begin{array}{c|cc} -0.7071 & 0 & 0.7071 \\ \hline 0 & 1.0000 & 0 \\ 0.7071 & 0 & 0.7071 \end{array} \right) = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$$

Step 2.

$$P_{DARE}^{EC} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -2 & -1 & 0 \\ -2 & -4 & 0 & -1 \end{pmatrix}$$

$$N_{DARE}^{EC} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.7071 & 0 & 0.7071 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Step 3.

$$Z = \begin{pmatrix} 0.8615 & -0.2781 & 0.3731 & -0.2034 \\ -0.3290 & 0.3256 & 0.8231 & -0.3290 \\ 0.2034 & 0.3731 & 0.2781 & 0.8615 \\ 0.3290 & 0.8231 & -0.3256 & 0.9329 \end{pmatrix}$$

Step 4.

$$X = \begin{pmatrix} 1.0000 & 2.0000 \\ 2.0000 & 4.2361 \end{pmatrix}$$

Verify: The residual norm = 7.772×10^{-16} .

MATLAB Note. MATLAB functions **care** and **dare** solve the CARE and DARE, respectively, using generalized Schur methods.

13.15.3 The Matrix Sign Function Methods

Let A be an $n \times n$ matrix with **no zero or purely imaginary eigenvalues**.

Let

$$J = X^{-1}AX = D + N,$$

where $D = \text{diag}(d_1, \dots, d_n)$ and N is nilpotent and commutes with D , be the Jordan Canonical Form of A . Then the matrix sign-function of A is defined as:

$\text{Sign}(A) = X \text{diag}(\text{sign}(d_1), \text{sign}(d_2), \dots, \text{sign}(d_n)) X^{-1}$, where

$$\text{sign}(d_i) = \begin{cases} 1 & \text{if } \text{Re}(d_i) > 0 \\ -1 & \text{if } \text{Re}(d_i) < 0 \end{cases}$$

Some important properties of $\text{Sign}(A)$ are [Exercise 17]:

1. $\text{Sign}(A)$ has the same stable invariant subspace as A .
2. The eigenvalues of $\text{Sign}(A)$ are ± 1 , depending upon the sign of the corresponding eigenvalues of A .
3. The range of $\text{Sign}(A) - I$ is the stable invariant subspace of A .
4. The eigenvectors of $\text{Sign}(A)$ are the eigenvectors and principal vectors of A .
5. $\text{Sign}(TAT^{-1}) = T\text{Sign}(A)T^{-1}$.

In this section, we will describe algorithms for solving the CARE and the DARE using matrix sign functions. Before doing so, let's first describe an algorithm for computing $\text{Sign}(A)$.

The basic sign-function algorithm is:

$$\begin{aligned} Z_0 &= A \\ Z_{k+1} &= \frac{1}{2} (Z_k + Z_k^{-1}), \quad k = 0, 1, \dots \end{aligned}$$

It can be shown that the sequence $\{Z_k\}$ converges to $\text{Sign}(A)$ quadratically.

The initial convergence can, however, be very slow. Byers (1987) has shown that the convergence can be accelerated if Z_k is scaled by $|\det(Z_k)|^{1/n}$. For a discussion of scaling see Kenney and Laub (1992).

Thus a **practical algorithm** for computing $\text{Sign}(A)$ is:

Algorithm 13.15.5 Computing $\text{Sign}(A)$.

Input: An $n \times n$ matrix A

Output: $\text{Sign}(A)$, the matrix sign-function of A .

Step 1. Set $Z_0 = A$

Step 2. For $k = 0, 1, 2, \dots$, do until convergence

Compute $c = |\det Z_k|^{1/n}$.

Compute $Z_{k+1} = \frac{1}{2c} (Z_k + c^2 Z_k^{-1})$

End

Stopping criteria. The algorithm can be terminated if

- (i) the norm of the difference between two successive iterates is small enough
or
- (ii) the number of iterations exceeds the maximum number prescribed.

The Matrix Sign-Function Method for the CARE

The mathematical basis for the matrix sign-function method for the CARE is the following theorem.

Theorem 13.15.7 (Roberts (1971)). Let H be the Hamiltonian matrix (13.2.1) associated with the CARE: $XA + A^T X + Q - X S X = 0$.

Let (A, B) be stabilizable and let (A, Q) be detectable.

Let $\text{Sign}(H) = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$, where W_{ij} are $n \times n$ real matrices.

Then the unique symmetric positive semidefinite stabilizing solution X of the CARE is given by the following **overdetermined** consistent linear systems:

$$\begin{pmatrix} W_{12} \\ W_{22} + I \end{pmatrix} X = - \begin{pmatrix} W_{11} + I \\ W_{21} \end{pmatrix}.$$

Proof:

Define

$$T = \begin{pmatrix} I & Y \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ -X & I \end{pmatrix} = \begin{pmatrix} I - YX & Y \\ -X & I \end{pmatrix},$$

where Y satisfies

$$(A - SX)Y + Y(A - SX)^T = -S.$$

An easy computation then shows that

$$THT^{-1} = \begin{pmatrix} A - SX & 0 \\ 0 & -(A - SX)^T \end{pmatrix}.$$

Note that $T^{-1} = \begin{pmatrix} I & -Y \\ X & I - XY \end{pmatrix}$.

Then, using Property 5 of the Sign-Function matrix we obtain

$$\begin{aligned} \text{Sign}(H) &= T^{-1} \text{Sign} \begin{pmatrix} A - SX & 0 \\ 0 & -(A - SX)^T \end{pmatrix} T \\ &= T^{-1} \begin{pmatrix} -I & 0 \\ 0 & I \end{pmatrix} T \quad (\text{since } A - SX \text{ is asymptotically stable}) \\ &= \begin{pmatrix} 2YX - I & -2Y \\ 2XYX - 2X & I - 2XY \end{pmatrix}. \end{aligned}$$

Thus $\text{Sign}(H) + I_{2n} = \begin{pmatrix} 2YX & -2Y \\ 2XYX - 2X & 2I - 2XY \end{pmatrix}$

or $\begin{pmatrix} W_{11} + I & W_{12} \\ W_{21} & W_{22} + I \end{pmatrix} = \left(\begin{bmatrix} 2Y \\ 2(XY - I) \end{bmatrix} X, -\begin{bmatrix} 2Y \\ 2(XY - I) \end{bmatrix} \right)$.

Comparing now both sides of the equation, we see that X must satisfy:

$$\begin{pmatrix} W_{12} \\ W_{22} + I \end{pmatrix} X = -\begin{pmatrix} W_{11} + I \\ W_{21} \end{pmatrix}.$$

■

Symmetric Version of the Matrix Sign-Function Algorithm

Theorem 13.5.7 yields a computational method to solve the CARE. However, the convergence can be painfully slow. The method can be made more efficient by using the following trick (Bierman (1984), Byers (1987)) in which one works only with symmetric matrices.

Define

$$W_0 = JH = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} = \begin{pmatrix} -Q & -A^T \\ -A & S \end{pmatrix}.$$

The matrix W_0 is symmetric.

Now compute $Sign(H)$ by performing the following iterations:

$$W_{k+1} = \frac{1}{2c_k} (W_k + c_k^2 JW_k^{-1} J), k = 0, 1, 2, \dots$$

Then each W_k is symmetric and $\lim_{k \rightarrow \infty} W_k = Jsign(H)$.

The parameter c_k is chosen to enhance the rate of convergence, as before.

Let $JSign(H) = Y = \begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix}$. Then

$$Sign(H) = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} = J^T Y,$$

The equations

$$\begin{pmatrix} W_{12} \\ W_{22} + I \end{pmatrix} X = -\begin{pmatrix} W_{11} + I \\ W_{21} \end{pmatrix}$$

then become

$$\begin{pmatrix} Y_{22} \\ Y_{12} + I \end{pmatrix} X = \begin{pmatrix} I - Y_{21} \\ -Y_{11} \end{pmatrix}.$$

This leads to the following symmetric version of the matrix sign-function algorithm for the CARE:

Algorithm 13.15.6 The Matrix Sign-Function Algorithm for the CARE.

Inputs: A - The $n \times n$ state matrix
 B - The $n \times m$ input matrix
 Q - The $n \times n$ state weighting matrix
 R - The $m \times m$ control weighting matrix
 ϵ - Error tolerance

Output: X - The symmetric positive semidefinite stabilizing solution of the CARE:

$$A^T X + X A - X B R^{-1} B^T X + Q = 0$$

Assumptions: 1) (A, B) is stabilizable and (A, Q) is detectable.
2) $Q \geq 0, R > 0$.

Step 1. 1.1 Form $S = BR^{-1}B^T$

1.2 Define $J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$. Form $W = JH = \begin{pmatrix} -Q & -A^T \\ -A & S \end{pmatrix}$

Step 2. For $k = 1, 2, \dots$ do until convergence with the given tolerance ϵ

$$c = |\det W|^{1/2n}$$

$$W = \frac{1}{2c}(W + c^2 JW^{-1}J)$$

Step 3. Partition $W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$, where each W_{ij} is of order n .

Step 4. Form $M = \begin{pmatrix} W_{22} \\ W_{12} + I_n \end{pmatrix}$, $N = \begin{pmatrix} I - W_{21} \\ -W_{11} \end{pmatrix}$.

Step 5. Solve for $X : MX = N$.

Example 13.15.5 Consider solving the CARE using Algorithm 13.5.6 with $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$,

$$B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, R = 1.$$

Step 1. $S = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$, $H = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 \end{pmatrix}$, $W_0 = JH = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Step 2. $W_1 = \frac{1}{2}(W_0 + JW_0^{-1}J) = \begin{pmatrix} -1 & 0 & 0 & -0.5 \\ 0 & -1 & -0.5 & 0 \\ 0 & -0.5 & 0.5 & 0 \\ -0.5 & 0 & 0 & 0.5 \end{pmatrix}$,

$$c = |\det(W_1)|^{\frac{1}{4}} = 0.8660$$

$$W_2 = \frac{1}{2c}(W_1 + c^2 JW_1^{-1}J) = \begin{pmatrix} -1.1547 & 0 & 0 & -0.5774 \\ 0 & -1.1548 & -0.5774 & 0 \\ 0 & -0.5774 & 0.5774 & 0 \\ -0.5774 & 0 & 0 & 0.5774 \end{pmatrix}.$$

(Note that each $W_i, i = 0, 1, 2$ is symmetric)

Step 3. $J \operatorname{Sign}(H) = W_2 = W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}$.

Step 5. $X = \begin{pmatrix} 1.7321 & 1 \\ 1 & 1.7321 \end{pmatrix}$.

Verify: The residual norm $= 9.9301 \times 10^{-16}$

Example 13.15.6 Consider now solving the CARE using Algorithm 13.5.6 with the following data:

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, R = 1.$$

Step 1.

$$W_0 = \begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & -1 & 0 & 3 \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 1 & 1 & 1 \\ 0 & 0 & 3 & 1 & 1 & 1 \end{pmatrix}$$

Step 2. After five iterations, $\| W_5 - W_4 \| / \| W_4 \| = 6.4200 \times 10^{-15}$ (The readers are asked to verify this by carrying out 5 iterations).

Step 3.

$$W \equiv W_5$$

Step 5.

$$X = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Verify: The residual norm $= 3.1602 \times 10^{-16}$.

Flop-count and Stability. It can be shown that Algorithm 13.5.6 requires about $4n^3$ flops per iteration. **The algorithm is not stable in general (Byers (1986b)),** unless used with an iterative refinement technique such as Newton's method (see **Section 13.5.4**).

MATCONTROL Note. Algorithm 13.5.6 has been implemented in MATCONTROL function **ricsgnc**.

The Matrix Sign-Function Method for the DARE.

The matrix sign-function method for the CARE described in the previous section can now be applied to solve the DARE by converting the symplectic matrix M to the Hamiltonian matrix H using the bilinear transformation:

$$H = (M + I)^{-1} (M - I).$$

Because A needs to be nonsingular, the method is not applicable if A is singular, and is not numerically effective when A is ill-conditioned.

Avoiding Explicit Inversion of A

The explicit inversion of A , however, may be avoided, by using the following simple trick (Gardiner and Laub (1986)).

Write

$$M = N^{-1}P,$$

$$\text{where } N = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}, \text{ and } P = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}.$$

Then it can be shown that even if A is singular, the matrix $(P + N)$ is invertible and the matrix H can be expressed as $H = (P + N)^{-1}(P - N)$.

Algorithm 13.15.7 The Matrix Sign-Function Algorithm for the DARE.

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix

Q - The $n \times n$ state weighting matrix

R - The $m \times m$ control weighting matrix

Output: X - The unique symmetric positive semidefinite stabilizing solution X of the DARE:

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$

Assumptions: 1) (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable,
 2) $Q \geq 0, R > 0$.

Step 1. Form $S = BR^{-1}B^T$, $N = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}$, $P = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}$.

Step 2. Form $H = (P + N)^{-1}(P - N)$.

Step 3. Apply the matrix sign-function algorithm for the continuous-time algebraic Riccati equation (Algorithm 13.5.6) with H in Step 2.

Example 13.15.7 Consider solving the DARE using Algorithm 13.5.7 with $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$,

$$B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, R = 1.$$

$$\text{Step 1. } S = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}.$$

$$\text{Step 2. } H = \begin{pmatrix} -0.3333 & 0.6667 & -0.6667 & 0.6667 \\ -0.6667 & 0.3333 & 0.6667 & -0.6667 \\ -1.3333 & 0.6667 & 0.3333 & 0.6667 \\ 0.6667 & -1.3333 & -0.6667 & -0.3333 \end{pmatrix}$$

$$\text{Step 3. } X = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

Verify: The residual norm = 6.7195×10^{-16} .

MATCONTROL Note. Algorithm 13.5.7 has been implemented in MATCONTROL function **ricsgnd**.

13.15.4 Newton's Methods

Recall that the classical Newton's method for finding a root x of $f(x) = 0$ can be stated as follows:

- (i) Choose x_0 , an initial approximation to x .
- (ii) Generate a sequence of approximations $\{x_i\}$ defined by

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad i = 0, 1, 2, \dots \quad (13.15.11)$$

Then, whenever x_0 is chosen close enough to x , the sequence $\{x_i\}$ converges to the root x and the convergence is quadratic if $f'(x) \neq 0$. Newton's methods for the CARE and DARE can similarly be developed.

Newton's Method for the Continuous-time Algebraic Riccati Equation

Consider first the CARE: $XA + A^T X - XBR^{-1}B^T X + Q = 0$.

Starting from an initial approximate solution X_0 , the computed solutions are iteratively refined until convergence occurs; and this is done by solving a Lyapunov equation at each iteration. The way how the Lyapunov equations arise can be explained as follows. Write $X = X_0 + (X - X_0)$. Substituting this into the CARE we have

$$\begin{aligned} & (A - BR^{-1}B^T X_0)^T X + X(A - BR^{-1}B^T X_0) \\ &= -X_0 BR^{-1}B^T X_0 - Q + (X - X_0)BR^{-1}B^T(X - X_0) \end{aligned}$$

Assuming that $X - X_0$ is small (that is, the initial approximate solution is good), we can neglect the last term on the right hand side of the above equation. Thus we obtain the following Lyapunov equation for the next approximation X_1 :

$$(A - BR^{-1}B^T X_0)^T X_1 + X_1(A - BR^{-1}B^T X_0) = -X_0 BR^{-1}B^T X_0 - Q$$

Assuming that X_1 is a better approximation than X_0 (that is $\|X - X_1\| < \|X - X_0\|$), the process can be continued until the convergence occurs, if there is convergence.

The above discussion immediately suggests the following **Newton method for the CARE**: (Kleinman (1968)):

Step 1. Choose an initial approximation X_0 .

Step 2. Compute $\{X_k\}$ iteratively by solving the Lyapunov equation:

$$(A - SX_k)^T X_{k+1} + X_{k+1}(A - SX_k) = -X_k SX_k - Q, \quad k = 0, 1, 2, \dots,$$

where $S = BR^{-1}B^T$.

Step 3. Continue until and if convergence occurs.

Newton's method, as stated above, is not in the familiar form. However, the above steps can be easily reorganized to obtain Newton's method in the familiar form (see Benner (1997), Hammarling (1982) and Lancaster and Rodman (1995) for details).

To do this, let's define

$$R_C(X) = XA + A^T X - XSX + Q,$$

where $S = BR^{-1}B^T$.

Now, the Fréchet derivative of $R_C(X)$ is given by

$$R'_X(Z) := (A - SX)^T Z + Z(A - SX)$$

Thus, Newton's method for $R_C(X) = 0$ is

$$R'_{X_i}(\Delta_i) + R_C(X_i) = 0, i = 0, 1, 2, \dots$$

$$X_{i+1} = X_i + \Delta_i$$

The above observation leads to the following Newton algorithm for the CARE.

Algorithm 13.15.8 Newton's Method for the CARE

Inputs: A - The $n \times n$ state matrix

B - The $n \times m$ input matrix

Q - The $n \times n$ state weighting matrix

R - The $m \times m$ control weighting matrix

Output: The set $\{X_k\}$ converging to an approximate stabilizing solution matrix X of the CARE.

Assumptions: 1) (A, B) is stabilizable and (A, Q) is detectable.

2) $Q \geq 0$, $R > 0$.

Step 1. Set $S = BR^{-1}B^T$.

Step 2. Choose an initial approximate solution $X_0 = X_0^T$ such that $A - SX_0$ is stable.

Step 3. Construct the sequence of solutions $\{X_i\}$ as follows:

For $i = 0, 1, 2, \dots$ do until convergence occurs

3.1 Compute $A_i = A - SX_i$

3.2 Compute $R_C(X_i) = A^T X_i + X_i A + Q - X_i S X_i$

3.3 Solve the Lyapunov equation for Δ_i : $A_i^T \Delta_i + \Delta_i A_i + R_C(X_i) = 0$.

3.4 Compute $X_{i+1} = X_i + \Delta_i$

End

Remarks: The above form of Newton's method is usually known as **Newton's Method in incremental form**. This form has some computational advantages over that presented in the beginning of this section in the sense that, in general, more accurate answers can be expected. This is because, in the incremental form algorithm, we solve the Lyapunov equation for the increment Δ_i and not for the solution directly and therefore, the solution X_i will have more correct digits.

The proof of the following theorem can be found in Lancaster and Rodman (1995, page 232-233). It gives conditions under which the above iterates converge.

Theorem 13.15.8 (Convergence of Newton's Method for the CARE).

Let (A, B) be stabilizable and (A, Q) be detectable. Let X_0 be an approximate stabilizing solution and let X be a unique symmetric positive semidefinite stabilizing solution X of the CARE. Then the matrices A_i and X_i , $i = 0, 1, \dots$, constructed by the above algorithm are such that

- (i) All A_i are stable; that is, all iterates X_i are stabilizing.
- (ii) $0 \leq X \leq \dots \leq X_{i+1} \leq X_i \leq \dots \leq X_1$.
- (iii) $\lim_{i \rightarrow \infty} X_i = X$, where X is the unique symmetric positive-semidefinite stabilizing solution of the CARE.
- (iv) There exists a constant $c > 0$ such that $\|X_{i+1} - X\| \leq c \|X_i - X\|^2$, for $i \geq 1$; that is, the sequence $\{X_i\}$ converges quadratically.

■

Remark: Note that (ii) and (iv) are also true for $i = 0$, if X_0 is close enough to X . But, in general, they are not true for $i = 0$ (see the counter-example later).

Stopping Criterion

The following can be used as a stopping criterion.

Stop the iteration if

I. For a certain value of k and the prescribed *tolerance* ϵ

$$\frac{\|X_{k+1} - X_k\|_F}{\|X_k\|_F} \leq \epsilon,$$

Or

II. The number of iterations k exceeds a prescribed number N .

If a condition-number estimator for the CARE is available, then criterion I can be replaced by the following more appropriate stopping criterion: Stop the iteration if

$$\frac{\|X_{k+1} - X_k\|_F}{\|X_k\|_F} \leq \mu \kappa_{CARE}^E,$$

where κ_{CARE}^E denotes an estimate of the κ_{CARE} and μ is the machine precision.

Example 13.15.8 Consider solving the CARE using Newton's method (**Algorithm 13.5.8**) with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad R = 1.$$

Step 1. $S = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

Step 2. $X_0 = \begin{pmatrix} 0.4 & 0.1 & 0.1 \\ 0.1 & 0.3 & 0.0 \\ 0.1 & 0 & 0.2 \end{pmatrix}$

Step 3.

$$\Delta_0 = \begin{pmatrix} i = 0 \\ -0.0248 & -0.0302 & -0.0369 \\ -0.0302 & -0.0426 & 0.0103 \\ -0.0369 & 0.0103 & -0.0224 \end{pmatrix}$$

$$X_1 = X_0 + \Delta_0 = \begin{pmatrix} 0.3752 & 0.0698 & 0.0631 \\ 0.0698 & 0.2574 & 0.0103 \\ 0.0631 & 0.0103 & 0.1776 \end{pmatrix}$$

Relative Change: $\frac{\|X_1 - X_0\|}{\|X_0\|} = 0.1465$

$$i = 1.$$

$$\Delta_1 = \begin{pmatrix} -0.0020 & -0.0015 & -0.0010 \\ -0.0015 & -0.0011 & -0.0008 \\ -0.0010 & -0.0008 & -0.0005 \end{pmatrix}$$

$$X_2 = X_1 + \Delta_1 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\frac{\|X_2 - X_1\|}{\|X_1\|} = 0.0086.$

$$i = 2$$

$$\Delta_2 = 10^{-5} \begin{pmatrix} -0.4561 & -0.3864 & -0.2402 \\ -0.3864 & -0.3311 & -0.2034 \\ -0.2402 & -0.2034 & -0.1265 \end{pmatrix}$$

$$X_3 = X_2 + \Delta_2 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\frac{\|X_3 - X_2\|}{\|X_2\|} = 2.1709 \times 10^{-5}.$

MATHCONTROL Note. Algorithm 13.5.8 has been implemented in MATCONTROL function **ricnwtnc**.

Convergence: We know that there exist infinitely many X_0 for which $A - SX_0$ is stable. *The choice of proper X_0 is crucial.* If the initial solution matrix X_0 is not close enough to the true solution X , then, as in the case of scalar Newton's method, the convergence can be painfully slow. *The method even might converge to a nonstabilizing solution in the presence of round-off errors.* Things might go wrong even at the first step. To see this, let's consider the following example from Kenney, Laub and Wette (1990):

$$A = 0, B = Q = I, R = I.$$

The exact solution is $X = I$. Let $X_0 = \epsilon I$, where $\epsilon > 0$ is a small positive number. Then

$$A - BB^T X_0 = -\epsilon I$$

is stable for all $\epsilon > 0$ and the initial error is $\|X - X_0\| = 1 - \epsilon \cong 1$ for small ϵ . However,

$$X_1 = \frac{1 + \epsilon^2}{2\epsilon} I,$$

and

$$\|X - X_1\| \simeq \frac{1}{2\epsilon},$$

which is quite large. Thus, even though the errors at subsequent steps decreases, a large number of steps will be needed for the error made at the first step to die out.

Some conditions guaranteeing convergence from the first step on have been given by Kenney, Laub, and Wette (1990). This is stated in the following Theorem (**assuming that $R = I_{m \times m}$**).

Theorem 13.15.9 *Let X_0 be an initial approximation such that $A - BB^T X_0$ is stable and assume that $\|X - X_0\| < \frac{1}{3\|B\|^2\|\Omega^{-1}\|}$, where $\Omega(Z) = (A - BB^T X)^T Z + Z(A - BB^T X)$, then $\|X - X_1\| \leq \|X - X_0\|$, with equality only when $X_0 = X$.*

Flop-Count. Newton's method is iterative; therefore, an exact flop count cannot be given. However, if the Schur method is used to solve the Lyapunov equations at each iteration, then about $40n^3$ flops are needed per iteration.

Stability. Since the principal computational task in Newton's Method is the solution of a Lyapunov matrix equation at each iteration, **the method can be shown to be stable if a numerically stable method such as the Schur method is used to solve the Lyapunov equation.** Specifically, if \hat{X} is the computed solution obtained by Newton's method, then it can be shown (Petkov et al. (1991)) that

$$\frac{\|\hat{X} - X\|_F}{\|X\|_F} \leq \mu\kappa_{CARE},$$

where κ_{care} is the condition number of the CARE. **That is, the method does not introduce more errors than what is already inherent in the problem.**

Modified Newton's Methods

Several modifications of Newton's methods for the algebraic Riccati equations have been obtained in recent years (Benner and Byers (1994, 1998), Guo and Lancaster (1998), Guo (1998), etc.). We just state in the following the line search modification of Newton's method by Benner and Byers (1998).

Newton's Method With Line Search

The performance of Newton's method can be improved by using an optimization technique, called **line search**.

The idea is to take a Newton step at each iteration in the direction so that $\|R_C(X_{i+1})\|_F^2$ is minimized. Thus the iteration

$$X_{i+1} = X_i + \Delta_i$$

in Step 3 of Newton's method will be replaced by

$$X_{i+1} = X_i + t_i \Delta_i,$$

where t_i is a real scalar to be chosen so that $\|R_C(X_i + t_i \Delta_i)\|_F^2$ will be minimized.

This is equivalent to minimizing

$$\begin{aligned} f_i(t) &= \text{Trace}(R_C(X_i + t\Delta_i)^T R_C(X_i + t\Delta_i)) = \text{Trace}(R_C(X_i + t\Delta_i)^2) \\ &= \alpha_i(1-t)^2 - 2\beta_i(1-t)t^2 + \nu_i t^4 \end{aligned}$$

where

$$\begin{aligned} \alpha_i &= \text{Trace}(R_C(X_i)^2) \\ \beta_i &= \text{Trace}(R_C(X_i)V_i) \\ \nu_i &= \text{Trace}(V_i^2), \\ V_i &= \Delta_i S \Delta_i. \end{aligned}$$

It can be shown [see Benner (1997), Benner and Byers (1998)] that the function $f_i(t)$ has a local minimum at some value $t_i \in [0, 2]$.

We thus have the following modified Newton's algorithm.

Algorithm 13.15.9 *Newton's Method With Line Search for the CARE*

Inputs: Same as in Algorithm 13.5.8

Output: Same as in Algorithm 13.5.8

Assumptions: Same as in Algorithm 13.5.8.

Step 1. Same as in Algorithm 13.5.8.

Step 2. Same as in Algorithm 13.5.8.

Step 3. For $i = 0, 1, 2, \dots$ do until convergence occurs

3.1 Same as in Algorithm 13.5.8.

3.2 Same as in Algorithm 13.5.8.

3.3 Same as in Algorithm 13.5.8.

3.4 Compute $V_i = \Delta_i S \Delta_i$

3.5 Compute α_i, β_i , and ν_i of f_i as given above.

3.6 Compute $t_i \in [0, 2]$ such that $f_i(t_i) = \min_{t \in [0, 2]} f_i(t)$.

3.7 Compute $X_{i+1} = X_i + t_i \Delta_i$.

End.

Example 13.15.9 The input matrices A, B, Q , and R are the same as in Example 13.5.8.

$$\text{Step 1. } S = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\text{Step 2. } X_0 = \begin{pmatrix} 0.4 & 0.1 & 0.1 \\ 0.1 & 0.3 & 0 \\ 0.1 & 0 & 0.2 \end{pmatrix}$$

$$\text{Step 3. } i = 0 : \Delta_0 = \begin{pmatrix} -0.0248 & -0.0302 & -0.0369 \\ -0.0302 & -0.0426 & 0.0103 \\ -0.0369 & 0.0103 & -0.0224 \end{pmatrix}$$

$$\alpha_0 = 0.1761, \beta_0 = -0.0049, \gamma_0 = 2.1827 \times 10^{-4}, t_0 = 1.0286$$

$$X_1 = X_0 + t_0 \Delta_0 = \begin{pmatrix} 0.3745 & 0.0690 & 0.0620 \\ 0.0690 & 0.2562 & 0.0105 \\ 0.0620 & 0.0105 & 0.1770 \end{pmatrix}$$

Relative Change: $\| X_1 - X_0 \| / \| X_0 \| = 0.1507$

$$i = 1: \Delta_1 = \begin{pmatrix} -0.0012 & -0.0006 & 0.0000 \\ -0.0006 & 0.0001 & -0.0011 \\ 0.0000 & -0.0011 & 0.0001 \end{pmatrix}$$

$$\alpha_1 = 8.9482 \times 10^{-5}, \beta_1 = -4.2495 \times 10^{-8}, \gamma_1 = 4.9519 \times 10^{-11}, t_1 = 1.0005.$$

$$X_2 = X_1 + t_1 \Delta_1 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\| X_2 - X_1 \| / \| X_1 \| = 0.0038587$

$$i = 2: \Delta_2 = 10^{-6} \begin{pmatrix} -0.1677 & -0.4428 & -0.4062 \\ -0.4428 & -0.7620 & 0.1277 \\ -0.4062 & 0.1277 & -0.2505 \end{pmatrix}$$

$$\alpha_2 = -2.9393 \times 10^{-10}, \beta_2 = -1.0425 \times 10^{-17}, \gamma_2 = 6.1179 \times 10^{-24}, t_2 = 1.0000.$$

$$X_3 = X_2 + t_2 \Delta_2 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\| X_3 - X_2 \| / \| X_2 \| = 2.4025 \times 10^{-6}$

$$i = 3: \Delta_3 = 10^{-12} \begin{pmatrix} -0.1593 & -0.0972 & 0.0319 \\ -0.0972 & -0.0286 & -0.1791 \\ 0.00319 & -0.1791 & 0.0308 \end{pmatrix}$$

$$\alpha_3 = 2.4210 \times 10^{-24}, \beta_3 = -1.4550 \times 10^{-37}, \gamma_3 = 2.4612 \times 10^{-50}, t_3 = 1.0000.$$

$$X_4 = X_3 + t_3 \Delta_3 = \begin{pmatrix} 0.3732 & 0.0683 & 0.0620 \\ 0.0683 & 0.2563 & 0.0095 \\ 0.0620 & 0.0095 & 0.1770 \end{pmatrix}$$

Relative Change: $\| X_4 - X_3 \| / \| X_3 \| = 5.5392 \times 10^{-13}.$

Theorem 13.15.10 (Convergence of Newton's Method with Line Search for the CARE)

If (A, B) is a stabilizable pair, and if the step sizes t_i are bounded away from zero, then Newton's method with the line search (**Algorithm 13.5.9**) converges to the stabilizing solution.

■

Proof: See Benner and Byers (1998), Guo and Laub (2000).

Flop-count Algorithm 13.5.9 is slightly more expensive (about 8% to the cost of a Newton step) than Algorithm 13.5.8. However, one saves about one iteration step out of 15; often much more, but seldom less.

MATCONTROL Note. Algorithm 13.5.9 has been implemented in MATCONTROL function **ricnwslsc**.

Newton's method for the Discrete Algebraic Riccati Equation

Newton's method for the DARE

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0,$$

is analogous. It is based on successive solutions of **Stein equations (discrete-time Lyapunov equations)** associated with the discrete-time system. We state the algorithm below without detailed discussions. The algorithm was originally developed by Hewer (1971). See also Kleinman (1974).

Algorithm 13.15.10 Newton's Method for the DARE

Inputs:

- A - The $n \times n$ state matrix
- B - The $n \times m$ input matrix
- Q - The $n \times n$ state weighting matrix
- R - The $m \times m$ control weighting matrix

Output: The set $\{X_k\}$ converging to the unique symmetric positive semidefinite stabilizing solution X of the DARE:

$$R_D(X) = A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$

Assumptions:

- 1) (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable.
- 2) $Q \geq 0, R > 0$.

Step 1. Choose $X_0 = X_0^T$ such that $A - B(R + B^T X_0 B)^{-1} B^T X_0 A$ is a discrete-stable matrix; that is, it has all its eigenvalues inside the unit circle.

Step 2. For $i = 0, 1, 2, \dots$ do until convergence.

- 2.1 Compute $K_i = (R + B^T X_i B)^{-1} B^T X_i A$
 - 2.2 Compute $A_i = A - BK_i$
 - 2.3 Compute $R_D(X_i) = A^T X_i A - X_i + Q - A^T X_i B(R + B^T X_i B)^{-1} B^T X_i A$
 - 2.4 Solve the discrete-time Lyapunov equation (Stein equation) for Δ_i :

$$A_i^T \Delta_i A_i - \Delta_i + R_D(X_i) = 0$$
 - 2.5 Compute $X_{i+1} = X_i + \Delta_i$
- End

The following theorem gives conditions under which the sequence $\{X_i\}$ converges. The proof of this theorem can be found in Lancaster and Rodman (1995, pp. 308-310.)

Theorem 13.15.11 (Convergence of Newton's Method for the DARE)

Suppose that (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable. Let X_0 be a stabilizing approximate solution of the DARE. Then the matrices A_i and X_i , constructed by the above algorithm, are such that

- (i) All A_i are discrete-stable.
- (ii) $0 \leq X \leq \dots \leq X_{i+1} \leq X_i \leq \dots \leq X_1$
- (iii) $\lim_{i \rightarrow \infty} X_i = X$, where X is the unique symmetric positive-semidefinite discrete-stabilizing solution of the DARE.
- (iv) There exists a constant $c > 0$ such that $\|X_{i+1} - X\| \leq c \|X_i - X\|^2$, $i \geq 1$, that is, the sequence $\{X_i\}$ converges quadratically.

Stopping Criterion

The same stopping criteria as in the case of Newton's method for the CARE can be used.

Example 13.15.10 Consider solving the DARE using Algorithm 13.5.10 with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad R = 1, \quad Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Step 1. $X_0 = \begin{pmatrix} 1 & -5 & 10 \\ -5 & 1600 & -2000 \\ 10 & -2000 & 2700 \end{pmatrix}$

Step 2. $i = 0$

The eigenvalues of $A - B(R + B^T X_0 B)^{-1} B^T X_0 A$ are $-0.8831 \pm j0.2910, -0.0222$. Then X_0 is a discrete-stabilizing approximate solution of the DARE.

$$K_0 = \begin{pmatrix} -0.0192 & 2.6154 & -6.8077 \end{pmatrix}$$

$$A_0 = \begin{pmatrix} -0.9808 & -1.6154 & 7.8077 \\ 0.0192 & -4.6154 & 6.8077 \\ 0.0192 & -2.6154 & 3.8077 \end{pmatrix}$$

$$X_1 = 10^4 \begin{pmatrix} 0.0008 & -0.0137 & 0.0167 \\ -0.0137 & 0.6808 & -0.9486 \\ 0.0165 & -0.9486 & 1.3364 \end{pmatrix}$$

Relative Change: $\frac{\| X_1 - X_0 \|}{\| X_0 \|} = 3.7654$

$$i = 1$$

$$K_1 = \begin{pmatrix} -0.0301 & 4.4699 & -9.5368 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} -0.9699 & -3.4699 & 10.5368 \\ 0.0301 & -6.4699 & 9.5368 \\ 0.0301 & -4.4699 & 6.5368 \end{pmatrix}$$

$$X_2 = 10^3 \begin{pmatrix} 0.0067 & -0.0893 & 0.1029 \\ -0.0893 & 2.0297 & -2.5658 \\ 0.1029 & -2.5658 & 3.3125 \end{pmatrix}$$

Relative Change: $\frac{\| X_2 - X_1 \|}{\| X_1 \|} = 0.7364$

$$i = 2$$

$$K_2 = \begin{pmatrix} -0.0826 & 5.1737 & -10.2938 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} -0.9174 & -4.1737 & 11.2938 \\ 0.0826 & -7.1737 & 10.2938 \\ 0.0826 & -5.1737 & 7.2938 \end{pmatrix}$$

$$X_3 = 10^3 \begin{pmatrix} 0.0054 & -0.0670 & 0.0767 \\ -0.0670 & 1.6234 & -2.0796 \\ 0.0767 & -2.0796 & 2.7283 \end{pmatrix}$$

Relative Change: $\frac{\| X_3 - X_2 \|}{\| X_2 \|} = 0.1862.$

The relative changes continue to decrease from this step onwards.

$$X_7 = 10^3 \begin{bmatrix} 0.0053 & -0.0658 & -0.0751 \\ -0.0658 & 1.5943 & -2.0428 \\ 0.0751 & -2.0428 & 2.6817 \end{bmatrix}$$

For $i = 6$, the **relative change**: $\frac{\| X_7 - X_6 \|}{\| X_6 \|}$ is 2.3723×10^{-15} .

MATCONTROL Note. Algorithm 13.5.10 has been implemented in MATCONTROL function **ricnwtnd**.

Newton's Method with Line Search for the DARE

Algorithm 13.5.10 can be modified in a similar way as in case of the CARE to include the line search.

The function $f_i(t)$ to be minimized in this case is given by

$$f_i(t) = \alpha_i(1-t)^2 - 2\beta_i(1-t)t^2 + \gamma_i t^4,$$

where $\alpha_i = \text{Trace}(R_d(X_i)^2)$

$\beta_i = \text{Trace}(R_d(X_i)V_i)$

$\gamma_i = \text{Trace}(V_i^2)$

and $V_i = A_i^T \Delta_i B (R + B^T X_i B)^{-1} B^T \Delta_i A_i$

For details, see Benner (1997), Benner and Byers (1998).

Algorithm 13.15.11 Newton's Method with Line Search for the DARE

Inputs: Same as in Algorithm 13.5.10

Output: Same as in Algorithm 13.5.10

Assumptions: Same as in Algorithm 13.5.10

Step 1. Same as in Algorithm 13.5.10

Step 2. For $k = 0, 1, 2, \dots$ do until convergence

- 2.1 Same as in Algorithm 13.5.10
- 2.2 Same as in Algorithm 13.5.10
- 2.3 Same as in Algorithm 13.5.10
- 2.4 Same as in Algorithm 13.5.10
- 2.5 Compute $S_i = B(R + B^T X_i B)^{-1} B^T$
- 2.6 Compute $V_i = A_i^T \Delta_i S_i \Delta_i A_i$
- 2.7 Compute the coefficients α_i, β_i and γ_i of $f_i(t)$ as above

2.8 Compute $t_i \in [0, 2]$ such that $f_i(t_i) = \min_{t \in [0, 2]} f_i(t)$.

2.9 $X_{i+1} = X_i + t_i \Delta_i$.

End

Flop-Count The algorithm is again just slightly more expensive than Algorithm 13.5.10. The additional cost of forming V_i , the coefficients of f_i , a local minimizer t_i of f_i and scaling Δ_i by t_i is cheap as compared to $O(n^3)$ flops required for other computations.

Convergence: The line search procedure can sometimes significantly improve the convergence behavior of Newton's method. For details, see Benner (1997).

Example 13.15.11 Consider solving the DARE using Algorithm 13.5.11 with

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad R = 1.$$

$$\text{Step 1: } X_0 = \begin{pmatrix} 1 & -5 & 10 \\ -5 & 1600 & -2000 \\ 10 & -2000 & 2700 \end{pmatrix}.$$

$$\text{Step 2: } i = 0, \quad \Delta_0 = 10^4 \begin{pmatrix} 0.0007 & -0.0132 & 0.0157 \\ -0.0132 & 0.5208 & -0.7486 \\ 0.0157 & -0.7486 & 1.0664 \end{pmatrix},$$

$$\alpha_0 = 9.7240 \times 10^7, \quad \beta_0 = 5.5267 \times 10^8, \quad \gamma_0 = 3.1518 \times 10^9, \quad t_0 = 0.3402.$$

$$X_1 = X_0 + t_0 D_0 = 10^3 \begin{pmatrix} 0.0034 & -0.0500 & 0.0635 \\ -0.0500 & 3.3718 & -4.5471 \\ 0.0635 & -4.5471 & 6.3283 \end{pmatrix}$$

$$\text{Relative Change: } \|X_1 - X_0\| / \|X_0\| = 1.2812.$$

$$\text{Step 3: } i = 1, \quad \Delta_1 = 10^3 \begin{pmatrix} 0.0029 & -0.0405 & 0.0431 \\ -0.0405 & -1.1655 & 1.7233 \\ 0.0431 & 1.7233 & -2.6498 \end{pmatrix},$$

$$\alpha_1 = 1.1123 \times 10^7, \quad \beta_1 = 1.7963 \times 10^6, \quad \gamma_1 = 3.0428 \times 10^5, \quad t_1 = 0.8750.$$

$$X_2 = X_1 + t_1 \Delta_1 = 10^3 \begin{pmatrix} 0.0059 & -0.0854 & 0.1012 \\ -0.0854 & 2.3520 & -3.0392 \\ 0.1012 & -3.0392 & 4.0097 \end{pmatrix}$$

$$\text{Relative Change: } \|X_2 - X_1\| / \|X_1\| = 0.3438.$$

$$i = 2, \quad \Delta_2 = 10^{-3} \begin{pmatrix} -0.0006 & 0.0196 & -0.0261 \\ 0.0196 & -0.7570 & 0.9955 \\ -0.0261 & 0.9955 & -1.3267 \end{pmatrix},$$

$$\alpha_2 = 1.9251 \times 10^5, \beta_2 = -157.2798, \gamma_2 = 0.1551, t_2 = 1.0008$$

$$X_3 = X_2 + t_2 \Delta_2 = 10^3 \begin{pmatrix} 0.0053 & -0.0658 & 0.0751 \\ -0.0658 & 1.5944 & -2.0429 \\ 0.0751 & -2.0429 & 2.6819 \end{pmatrix}$$

$$\textbf{Relative Change: } \| X_3 - X_2 \| / \| X_2 \| = 0.3283$$

$$i = 3, \Delta_3 = \begin{pmatrix} -0.0003 & 0.0024 & -0.0011 \\ 0.0024 & -0.0481 & 0.1094 \\ -0.0011 & 0.1094 & -0.2202 \end{pmatrix},$$

$$\alpha_3 = 0.0912, \beta_3 = -2.8785 \times 10^{-5}, \gamma_3 = 1.6525 \times 10^{-8}, t_3 = 1.0003$$

$$X_4 = X_3 + t_3 \Delta_3 = 10^3 \begin{pmatrix} 0.0053 & -0.0658 & -0.0751 \\ -0.0658 & 1.5943 & -2.0428 \\ 0.0751 & -2.0428 & 2.6817 \end{pmatrix}$$

$$\textbf{Relative Change: } \| X_4 - X_3 \| / \| X_3 \| = 6.4273 \times 10^{-5}.$$

$$i = 4, \Delta_4 = 10^{-4} \begin{pmatrix} 0.0001 & 0.0182 & -0.0295 \\ 0.0182 & -0.5017 & 0.4913 \\ -0.0295 & 0.4913 & -0.3757 \end{pmatrix},$$

$$\alpha_4 = 7.4477 \times 10^{-9}, \beta_4 = -1.2874 \times 10^{-15}, \gamma_4 = 4.9961 \times 10^{-22}, t_4 = 1.0000$$

$$X_5 = X_4 + t_4 \Delta_4 = 10^3 \begin{pmatrix} 0.0053 & -0.0658 & 0.0751 \\ -0.0658 & 1.5943 & -2.0428 \\ 0.0751 & -2.0428 & 2.6817 \end{pmatrix}$$

$$\textbf{Relative Change: } \| X_5 - X_4 \| / \| X_4 \| = 2.1982 \times 10^{-8}.$$

$$i = 5, \Delta_5 = 10^{-9} \begin{pmatrix} -0.0001 & 0.0033 & -0.0042 \\ 0.0033 & -0.1537 & 0.2147 \\ -0.0043 & 0.2185 & -0.3126 \end{pmatrix},$$

$$\alpha_5 = 3.6928 \times 10^{-22}, \beta_5 = -1.4449 \times 10^{-34}, \gamma_5 = 2.6879 \times 10^{-46}, t_5 = 1.0000$$

$$X_6 = X_5 + t_5 \Delta_5 = 10^3 \begin{pmatrix} 0.0053 & -0.0658 & 0.0751 \\ -0.0658 & 1.5943 & -2.0428 \\ 0.0751 & -2.0428 & 2.6817 \end{pmatrix}$$

$$\textbf{Relative Change: } \| X_6 - X_5 \| / \| X_5 \| = 1.0906 \times 10^{-13}$$

$$\textbf{Relative Residual} = 3.2312 \times 10^{-11}.$$

MATCONTROL Note: Algorithm 13.5.11 has been implemented in MATCONTROL function `ricnwlsd`.

Newton's Method as an Iterative Refinement Technique

Newton's method is often used as an **iterative refinement technique**. First, a direct robust method such as the Schur method or the matrix sign function method is applied to obtain an approximate solution and this approximate solution is then refined using a few iterative steps of Newton's method. **For higher efficiency, Newton's method with the line search (Algorithm 13.5.9 for the CARE and Algorithm 13.5.11 for the DARE) should be preferred over Newton's method.**

13.16 The Schur and Inverse-Free Generalized Schur Methods for the Descriptor Riccati Equations

As we have seen in Chapter 5 that several practical applications give rise to the descriptor systems:

$$E\dot{x}(t) = Ax(t) + Bu(t) \quad (\text{Continuous-time}) \quad (13.16.1)$$

$$Ex_{k+1} = Ax_k + Bu_k \quad (\text{Discrete-time}) \quad (13.16.2)$$

If E is nonsingular, then the algebraic Riccati equations **associated** with the above systems, respectively, are:

$$A^T X E + E^T X A - E^T X B R^{-1} B^T X E + Q = 0, \quad (13.16.3)$$

and

$$E^T X E = A^T X A - A^T X B (B^T X B + R)^{-1} B^T X A + Q. \quad (13.16.4)$$

The Riccati equations (13.6.3) and (13.6.4) will be, respectively, called as the **descriptor continuous-time algebraic Riccati equation** (DCARE) and the **descriptor discrete-time algebraic Riccati equation** (DDARE).

Most of the methods, such as the Schur method, the matrix sign function method, and Newton's method, can be easily extended to solve DCARE and DDARE.

Below we state how the generalized Schur methods and the inverse-free generalized Schur methods can be extended to solve these equations. The derivations of the others are left as **Exercises**. See Bender and Laub (1985, 1987), Benner (1997), Laub (1991), Mehrmann (1988), etc. in this context.

13.16.1 The Generalized Schur Method for the DCARE

The matrix pencil associated with the DCARE is

$$P_{DCARE} - \lambda N_{DCARE} = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix} - \lambda \begin{pmatrix} E & O \\ O & E^T \end{pmatrix},$$

where $S = BR^{-1}B^T$.

The Schur method for the DCARE, then, can be easily developed by transforming the above pencil to the **Ordered real Schur** form using the QZ iteration algorithm (Chapter 4). Thus, if Q_1 and Z_1 are orthogonal matrices such that

$$Q_1 P_{DCARE} Z_1 = \begin{pmatrix} L_{11} & L_{12} \\ O & L_{22} \end{pmatrix}, \quad Q_1 N_{DCARE} Z_1 = \begin{pmatrix} N_{11} & N_{12} \\ O & N_{22} \end{pmatrix},$$

where $Q_1 P_{DCARE} Z_1$ is upper quasi triangular, $Q_1 N_{DCARE} Z_1$ is upper triangular, and $L_{11} - \lambda N_{11}$ is stable; then the columns of $\begin{pmatrix} Z_{11} \\ Z_{21} \end{pmatrix}$, where $Z_1 = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$, span the stable deflating subspace. So, the matrix $X = Z_{21} Z_{11}^{-1}$ is a solution of the DCARE.

MATLAB Note. MATLAB function **care** in the form

$$[X, L, G, rr] = \text{care}(A, B, Q, R, E)$$

solves the DCARE.

Here $G = R^{-1}(B^T X E)$, the gain matrix, $L = \text{eig}(A - BG, E)$ and rr = the Frobenius norm of the relative residual matrix.

13.16.2 The Inverse-Free Generalized Schur Method for the DCARE

In case R is singular or nearly singular, one needs to use the inverse-free generalized Schur method. The extended pencil to be considered in this case is

$$\begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix} - \lambda \begin{pmatrix} E & 0 & 0 \\ 0 & E^T & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

This extended pencil is then compressed into an $2n \times 2n$ pencil in the same way as in Algorithm 13.5.3 and the rest of the procedure is the same as that algorithm.

13.16.3 The Inverse-Free Generalized Schur Method for the DDARE

The matrix pencil associated with the DDARE is

$$\begin{pmatrix} A & 0 \\ -Q & E^T \end{pmatrix} - \lambda \begin{pmatrix} E & S \\ 0 & A^T \end{pmatrix}, \text{ where } S = BR^{-1}B^T.$$

The extended pencil for the **Inverse-free generalized Schur method for the DDARE** is

$$\begin{pmatrix} A & 0 & -B \\ -Q & E^T & 0 \\ 0 & 0 & R \end{pmatrix} - \lambda \begin{pmatrix} E & 0 & 0 \\ 0 & A^T & 0 \\ 0 & B^T & 0 \end{pmatrix}.$$

The pencil is now compressed into an $2n \times 2n$ pencil as in Algorithm 13.5.4 and the rest of the steps of Algorithm 13.5.4 is then followed.

MATLAB Note. The MATLAB function **dare** in the form $[X, L, G, rr] = \text{dare}(A, B, Q, R, E)$ solves the DDARE. Here $G = (B^T X B + R)^{-1} B^T X A$, $L = \text{eig}(A - BG, E)$, and rr = the Frobenius norm of the relative residual matrix.

13.17 Conclusions and Table of Comparisons

In this section, we present a table of comparisons of different methods discussed in this chapter and give a guideline for practical uses of these methods, based on this comparative study. We only present the table below for the CARE. A similar table can be set up for the DARE as well. However, the comments made about the Schur method for the CARE are not valid for the DARE; because *the Schur method for the DARE does not work when A is singular and is expected to give inaccurate results when A is theoretically nonsingular, but is computationally nearly singular.*

A Table of Comparisons of Different Methods for the CARE

Method	Efficiency, Stability, and Convergence Properties	Remarks
The Eigenvector and the Generalized Eigenvector Methods	The methods are in general not numerically stable (They become unstable when the Hamiltonian matrix has nearly multiple eigenvalues).	Not recommended to be used in practice.
The Schur Method	Stable in practice.	Widely used.
The Symplectic Hamiltonian-Schur Method	Stable and Structure-Preserving. Requires less computations and storage for problems of size greater than 20.	Works in the single-input and/or single-output case.
The Extended Hamiltonian-Schur Method	Stable and Structure-Preserving. More-efficient than the Schur-method.	Works in the multi-input case.
Newton's Method	Convergence is quadratic if the initial approximation is close to the solution. The convergence can be painfully slow if the initial approximation is far from the solution.	Usually used as an iterative refinement procedure .
The Matrix Sign-Function Method	Not Stable in general. Though iterative in nature; unlike Newton's Method, does not require the knowledge of a stabilizing initial guess.	Simple to use and is structure preserving. Recommended to be used in conjunction with Newton's method.
The Generalized Schur Method	Stable in practice.	Does not work if the control weighting matrix R is singular. Even if R is theoretically nonsingular, the method should not be used if it is ill-conditioned.
The Inverse-Free Generalized Schur Method	Stable in practice	The best way to solve the CARE when R is nearly singular.

Conclusions and Recommendations: In conclusion, the following recommendations are made: **For the CARE:** *The Schur method (**Algorithm 13.5.1**) or the matrix sign function (**Algorithm 13.5.6**) method followed by Newton's iteration with line search (**Algorithm 13.5.9**) is recommended. If R is singular or nearly singular, then the inverse-free generalized Schur method (**Algorithm 13.5.3**) should be used in place of the Schur method or the matrix sign function method.*

For the DARE: *The inverse-free generalized Schur method (**Algorithm 13.5.4**) or the matrix sign function method (**Algorithm 13.5.7**) followed by Newton's method with line search (**Algorithm 13.5.10**) is recommended. However, the matrix sign function method should be avoided if R is nearly singular.*

13.18 Some Selected Software

13.18.1 MATLAB CONTROL SYSTEM TOOLBOX

Matrix equation solvers.

- care - Solve continuous algebraic Riccati equations
- dare - Solve discrete algebraic Riccati equations

13.18.2 MATCONTROL

- | | |
|----------|---|
| RICEIGC | - The eigenvector method for the continuous-time Riccati equation |
| RICSCHC | - The Schur method for the continuous-time Riccati equation |
| RICSCHD | - The Schur method for the discrete-time Riccati equation |
| RICGEIGD | - The generalized eigenvector method for the discrete-time Riccati equation |
| RICNWTNC | - Newton's method for the continuous-time Riccati equation |
| RICNWTND | - Newton's method for the discrete-time Riccati equation |
| RICSGNC | - The matrix sign-function method for the continuous-time Riccati equation |
| RICSGND | - The matrix sign-function method for the discrete-time Riccati equation |
| RICNWLS | - Newton's method with line search for the continuous-time Riccati equation |
| RICNWLD | - Newton's method with line search for the discrete-time Riccati equation |

13.18.3 CSP-ANM

Solutions of the algebraic Riccati equations

- The Schur method is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → SchurDecomposition]` (continuous-time case) and `DiscreteRiccatiSolve [a, b, q, r, SolveMethod → SchurDecomposition]` (**discrete-time case**).
- Newton's method is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → Newton, InitialGuess → w0]` (discrete-time case).
- The matrix Sign-function method is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → MatrixSign]` (continuous-time case) and `DiscreteRiccatiSolve [a, b, q, r, SolveMethod → MatrixSign]` (discrete-time case).
- The inverse-free method based on generalized eigenvectors is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → GeneralizedEigendecomposition]` (continuous-time case) and `DiscreteRiccatiSolve [a, b, q, r, SolveMethod → GeneralizedEigendecomposition]` (discrete-time case).
- The inverse-free method based on generalized Schur decomposition is implemented as `RiccatiSolve [a, b, q, r, SolveMethod → GeneralizedSchurDecomposition]` (continuous-time case) and `DiscreteRiccatiSolve [a, b, q, r, SolveMethod → GeneralizedSchurDecomposition]` (discrete-time case).

13.18.4 SLICOT

Riccati Equations

- SB02MD Solution of algebraic Riccati equations (Schur vectors method)
- SB02MT Conversion of problems with coupling terms to standard problems
- SB02ND Optimal state feedback matrix for an optimal control problem
- SB02OD Solution of algebraic Riccati equations (generalized Schur method)
- SB02PD Solution of continuous algebraic Riccati equations (matrix sign function method) with condition and forward error bound estimates
- SB02QD Condition and forward error for continuous Riccati equation solution
- SB02RD Solution of algebraic Riccati equations (refined Schur vectors method) with condition and forward error bound estimates
- SB02SD Condition and forward error for discrete Riccati equation solution

13.18.5 MATRIX_X

Purpose: Solve Riccati equation. Using the option ‘DISC’ solves the discrete Riccati equation.

Syntax: [EV, KC]=RICCATI (S, Q, NS, ‘DISC’)
[EV, KC, P]=RICCATI (S, Q, NS, ‘DISC’)

Purpose: Solves the indefinite Algebraic Riccati Equation (ARE): $A'P + PA - PRP + Q = 0$

Syntax: [P, SOLSTAT]=SINGRICKATI (A, Q, R { ,TYPE})

13.19 Summary and Review

As we have seen in Chapter 10 and Chapter 12 that the algebraic Riccati equations

$$XA + A^T X - XBR^{-1}B^T X + Q = 0 \text{ (CARE)}$$

and

$$A^T XA - X + Q - A^T XB(R + B^T XB)^{-1}B^T XA = 0 \text{ (DARE)}$$

arise in many areas of control systems design and analysis, such as:

- The LQR and LQG Designs
- Optimal State Estimation (Kalman Filter)
- H_∞ -Control
- Spectral Factorizations (not described in this book, see Van Dooren (1981)).

I. Existence and Uniqueness of Stabilizing Solution. Let $Q \geq 0$ and $R > 0$. If (A, B) is stabilizable and (A, Q) is detectable, then the CARE admits a unique symmetric positive semidefinite stabilizing solution (**Theorem 13.2.6**).

Such a solution is given by $X = X_2 X_1^{-1}$, where the columns of the matrix $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ span the stable invariant subspace of the **Hamiltonian matrix** $H = \begin{pmatrix} A & -S \\ -Q & -A^T \end{pmatrix}$, where $S = BR^{-1}B^T$.

An analogous result for the DARE also exists (**Theorem 13.3.2**). In this case the symplectic matrix

$$M = \begin{pmatrix} A + S(A^{-1})^T Q & -S(A^{-1})^T \\ (-A^{-1})^T Q & (A^{-1})^T \end{pmatrix}$$

takes the role of the Hamiltonian matrix.

II. Conditioning of the Riccati Equations

The absolute and the relative condition numbers of the CARE have been identified using a perturbation result (**Theorem 13.4.1**).

An approximate condition number of the CARE, using a first-order estimate is Byers' condition number (in **Frobenius norm**):

$$\kappa_{CARE}^B = \frac{\|\Omega^{-1}\| \|Q\| + \|\Theta\| \|A\| + \|\Pi\| \|S\|}{\|X\|},$$

where X is the stabilizing solution of the CARE and Ω , Π , and Θ are defined by:

$$\begin{aligned} \Omega(Z) &= (A - SX)^T Z + Z(A - SX), \\ \Theta(Z) &= \Omega^{-1}(Z^T X + XZ), \\ \Pi(Z) &= \Omega^{-1}(XZX), \\ \|\Omega^{-1}\|_F &= \frac{1}{\text{sep}(A_C^T, -A_C)}, \end{aligned}$$

where $A_C = A - SX$, $S = BR^{-1}B^T$.

The quantities $\|\Omega^{-1}\|$, $\|\Theta\|$, and $\|\Pi\|$ are computationally intensive. However lower and upper bounds of κ_{CARE}^B can be obtained by solving the following Lyapunov equations:

$$(A - SX)^T H_k + H_k(A - SX) = -X^k, k = 0, 1, 2$$

The large norms of these matrices (relative to the stabilizing solution X), in general, indicate that the CARE is ill-conditioned.

The condition number of the DARE is given by (13.4.16).

A first-order estimator for the condition number of the DARE is

$$\kappa_{DARE}^E = \frac{2 \| A \|_F^2 \frac{\| Q \|_F}{\| X \|_F} + \| A \|_F^2 \| S \|_F \| X \|_F}{sep(A_d^T, A_d)},$$

where $A_d = A - B(R + B^T X B)^{-1} B^T X A$, $S = BR^{-1}B^T$. The quantity $sep(A_d^T, A_d)$ can be determined as the minimum singular value of the matrix $A_d^T \otimes A_d^T - I_n^2$.

III. Numerical Methods for the Riccati Equations

The existing numerical methods for the Riccati equations can be broadly classified into the following four classes:

- Invariant Subspace Methods
- Deflating Subspace Methods
- The Matrix Sign Function Methods
- Newton's Methods

A basic idea of finding a stabilizing solution of the CARE (DARE), using an invariant subspace method, is to construct a basis for the stable invariant subspace of the Hamiltonian matrix H (symplectic matrix M). Such a basis can be constructed by using the eigenvectors or the Schur vectors of the Hamiltonian matrix H (the symplectic matrix M). The eigenvector matrix can be ill-conditioned if the matrix H (the matrix M) is nearly defective; and, therefore, **the eigenvector approach is not recommended to be used in practice.** The Schur method is preferable to the eigenvector method. If $U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$ is the **ordered Real Schur form** of H , and the eigenvalues with negative real parts are contained in T_{11} , then $X = U_{21} U_{11}^{-1}$ is the stabilizing solution of the CARE, where $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$.

The Schur method for the DARE can be similarly developed by finding an ordered real Schur form of the symplectic matrix M . However, since computation of the matrix M requires the explicit inversion of A , **the Schur method for the DARE does not work if A is singular or can be problematic if A is theoretically nonsingular but is computationally singular.** In such cases, a deflating subspace method should be used.

The idea behind a deflating subspace method is basically the same as that of an invariant subspace method except that the solution of the Riccati equation is now found by computing a basis for the deflating subspace of a matrix pencil. For the CARE, the pencil in $P_{CARE} - \lambda N_{CARE}$, where $P_{CARE} = \begin{pmatrix} A & -S \\ -Q & A^T \end{pmatrix}$, $N_{CARE} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$. For the

DARE, the matrices of the pencil are

$$P_{DARE} = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}, \quad N_{DARE} = \begin{pmatrix} I & S \\ 0 & A^T \end{pmatrix}.$$

Again, for reasons stated above, the **generalized Schur decomposition using the QZ algorithm should be used to compute such a basis. See Section 13.5.2 for details. The eigenvector approach should be avoided.**

Both the Schur methods and the generalized Schur methods require an explicit inversion of the matrix R . In case R is ill-conditioned with respect to matrix inversion, these methods may not give accurate solutions. The difficulties can be overcome by using an extended $(2n + m) \times (2n + m)$ pencil.

For the CARE, the extended pencil is $P_{CARE}^E - \lambda N_{CARE}^E$, where

$$P_{CARE}^E = \begin{pmatrix} A & 0 & B \\ -Q & -A^T & 0 \\ 0 & B^T & R \end{pmatrix},$$

and

$$N_{CARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

This extended $(2n + m) \times (2n + m)$ pencil can then be compressed into a $2n \times 2n$ pencil by finding the QR factorization of $\begin{pmatrix} R \\ B \end{pmatrix}$, without affecting the deflating subspace. The solution of the CARE then can be obtained by finding the ordered generalized Schur form of the compressed pencil.

For the DARE, the extended pencil is $M_{DARE}^E - \lambda N_{DARE}^E$, where

$$P_{DARE}^E = \begin{pmatrix} A & 0 & -B \\ -Q & I & 0 \\ 0 & 0 & R \end{pmatrix},$$

and

$$N_{DARE}^E = \begin{pmatrix} I & 0 & 0 \\ 0 & A^T & 0 \\ 0 & B^T & 0 \end{pmatrix}.$$

This $(2n + m) \times (2n + m)$ can be compressed into an $2n \times 2n$ pencil by using the QR factorization of $\begin{pmatrix} R \\ -B \end{pmatrix}$. For details, see **Section 13.5.2**.

Again, the required basis should be constructed by finding the generalized real Schur form of the pencil using the QZ algorithm.

13.20 Chapter Notes and Further Reading

The algebraic Riccati equations have been very well studied in the literatures of mathematics and control and filter theory.

For an excellent account of up-to-date theoretical developments, see the recent book of Lancaster and Rodman (1995). Some of the earlier theoretical developments are contained in Kučera (1972, 1979), Coppel (1974), and Singer and Hammarling (1983), Willems (1971), Wimmer (1984, 1994). The books by Anderson and Moore (1990), Ando (1988), Kwakernaak and Sivan (1972), Kimura (1997), Zhou et al. (1996) also contain a fair amount of theory of algebraic Riccati equations. The existence of maximal solutions for generalized algebraic Riccati equations arising in stochastic control has been discussed in DeSouza and Fragoso (1990). The paper by DeSouza, Gevers and Goodwin (1986) deals with Riccati equations arising in optimal filtering of nonstabilizable systems having singular state transition matrices.

Important numerical methods have been dealt with in details in the books by Sima (1996) and Mehrmann (1991). Benner (1999) has given an up-to-date review with special attention to structure-preserving methods. An extensive bibliography on numerical methods appear in Laub (1991), and Benner (1997). See Jamshidi (1980) for an earlier review.

For a review of periodic Riccati equations see the article of Bittanti et al. and the references therein in the book “**The Riccati Equation**” edited by Bittanti, Laub and Willems (1991). The latter contains several important papers on Riccati equations and the paper by Bittanti gives a brief life history of Count Jacopo Riccati (1676-1754), which is certainly worth reading. The sensitivity of the continuous-time Riccati equations has been studied by several people: Byers (1985), Kenney and Hewer (1990), Chen (1988), Konstantinov et.al. (1990), Xu (1996), Sun (1998), and Ghavimi and Laub (1995). **Theorem 13.4.1** is due to Sun (1998). The bound (13.4.14) is due to Kenney and Hewer (1990). The residual of an approximate stabilizing solution (**Theorem 13.4.3**) is due to Sun (1997a). The sensitivity of the discrete-time algebraic Riccati equation has been studied in Gudmundsson, Kenney and Laub (1992), Konstantinov et. al. (1993), and Sun (1998). The paper by Ghavimi and Laub (1995) relates backward error and sensitivity to accuracy and discusses techniques for refinement of computed solutions of the algebraic Riccati equations.

The eigenvector methods for the Riccati equations were proposed by McFarlane (1963) and Potter (1966). The Schur method for the Riccati equations originally appeared in the famous paper by Laub (1979). Petkov, Christov, and Konstantinov (1987) studied the numerical properties of the Schur method and concluded that the Schur method can be unstable in some cases and the solutions may be inaccurate. A further analysis by Kenney, Laub, and Wette (1989) attributed such inaccuracy to poor scaling. For an excellent account of scaling of the Schur methods, see Benner (1997). The structure-preserving Hamiltonian-Schur method was first proposed by Byers in his Householder-prize winning Ph.D thesis (1983) in the case of a single-input system ($\text{rank}(B) = 1$). See Byers (1986a) for details of the method. The theoretical foundation of this method is contained in the well-known paper by Paige and Van Loan (1981). Their result was later extended to the case when the Hamiltonian matrix has eigenvalues on the

imaginary axis by Lin and Ho (1990). Patel, Lin and Misra (1994) have discussed computation of stable invariant subspaces of Hamiltonian matrices. Another method, called the multishift method to compute the invariant subspace of the Hamiltonian matrix corresponding to the stable eigenvalues, was developed by Ammar and Mehrman (1993). The algorithm is called multishift algorithm because n stable eigenvalues of the Hamiltonian matrix are used as shifts to isolate the desired invariant subspace. The multishift method sometimes has convergence problems, particularly for large n . The Hamiltonian-Schur algorithm in the multi-input case is due to Benner, Mehrmann and Xu (1997). A good account of the structure preserving eigenvalue methods appears in Bunse-Gerstner, Byers and Mehrmann (1992). Mehrmann (1988) has given a structure-preserving method for the discrete-time Riccati equation with single-input and single-output. The non-orthogonal symplectic methods have been discussed by Bunse-Gerstner and Mehrmann (1986) and Bunse-Gerstner, Mehrmann and Watkins (1989) for the CARE, and by Benner, Fassbender and Watkins (1999), Fassbender and Benner (2001) for the DARE. The details of these methods and other references can be found in the recent book by Fassbender (2000).

The generalized eigenvalue problem approach leading to deflating subspace method for the discrete-time Riccati equation was proposed in Pappas, Laub, and Sandell (1980). See also Arnold and Laub (1984), Emami-Naeini and Franklin (1979, 1982). The inverse-free methods (the extended pencil approach (**Algorithm 13.5.3** and **Algorithm 13.5.4**)) and the associated compressed techniques were proposed by Van Dooren (1981).

The idea of using matrix sign function to solve the CARE was first introduced by Roberts (1971, 1980). Byers (1986b, 1987) discussed numerical stability of the method and studied the computational aspects in details. See also Bierman (1984) and Bai and Demmel (1998). A generalization of the matrix sign function method to a matrix pencil and its application to the solutions of DCARE and DDARE was proposed by Gardiner and Laub (1986). For a summary of the matrix sign function, see the recent paper of Kenney and Laub (1995). For a perturbation analysis of the matrix sign function, see Sun (1997c). Howland (1983) relates matrix sign function to separation of matrix eigenvalues.

For details of Newton's algorithm for the continuous-time algebraic Riccati equation (**Algorithm 13.5.8**) and that for the discrete-time algebraic Riccati equation (**Algorithm 13.5.10**), as presented here, see Benner (1997), Lancaster and Rodman (1995). The correct proof of convergence of Newton's method (**Theorem 13.5.8**) seemed to appear for the first time in Lancaster and Rodman (1995).

Kenney, Laub, and Wette (1990) gave results on error bounds for Newton's method, where it was first pointed out that if the initial solution X_0 is not chosen carefully, the error on the first step can be disastrous. They also gave conditions which guarantee monotone convergence from the first step on (**Theorem 13.5.9**). Several modifications of Newton's methods have appeared in recent years (Guo (1998), Guo and Lancaster (1998), Guo and Laub (2000), etc.). The line search modification proposed by Benner and Byers (1998) is extremely useful in practice. In general, it improves the convergence behavior of Newton's method and avoids the problem of

a disastrously large first step.

Ghavimi, Kenney, and Laub (1992) have discussed the local convergence analysis of conjugate gradient methods for solving the algebraic Riccati equations.

For an account of parallel algorithms for algebraic Riccati equations, see Bai and Qian (1994), Gardiner and Laub (1991), and Laub (1991) and references therein, Quintana and Hernández (1996a, 1996b, 1996c), etc.

For large-scale solutions of the algebraic Riccati equations see Ferng, Lin, and Wang (1997), Lu and Lin (1993), Jaimoukha and Kasenally (1994) and Benner and Fassbender (1997). The recent book by Ionescu, Oara and Weiss (1999) gives a nice treatment of algebraic Riccati equations for the indefinite sign and singular cases. See also Campbell (1980). For least-squares solutions of stationary optimal control using the algebraic Riccati equations, see Willems (1971).

Some discussions on finding the Cholesky factor of the solution to an algebraic Riccati equation without first computing the solution itself appears in Singer and Hammarling (1983). Lin (1987) has given a numerical method for computing the closed-loop eigenvalues of a discrete-time Riccati equation. Patel (1993) has given a numerical method for computing the eigenvalues of a symplectic matrix. For numerical algorithms for descriptor Riccati equations, see Benner (1999), Mehrmann (1991), Bender and Laub (1985, 1987), Benner, Mehrmann and Xu (1999), etc. A description of discrete-time descriptor Riccati equations also appears in Zhang, Lam, and Zhang (1999). A comparative study with respect to efficiency and accuracy of most of the methods described in this chapter for the CARE (the **eigenvector, Schur, inverse-free generalized Schur, Hamiltonian-Schur and Newton's Methods**) has been made in the recent M.Sc Thesis of Ho (2000), using MATLAB and FORTRAN-77 codes (In particular, this thesis contains MATLAB codes for ordered **Real Schur** and **Generalized Real Schur** decompositions). Numerical experiments were performed on 12 benchmark examples taken from the collection of Benner, Laub, and Mehrmann (1995). The conclusions drawn in this thesis are almost identical to those mentioned in **Section 13.7**. For a recent collection of benchmark examples for Riccati equations, see Abels and Benner (1999a, 1999b).

EXERCISES

1. Prove that if (A, B) is controllable and (A, Q) is observable, then the CARE (13.1.1) admits a unique symmetric positive definite solution X .
2. Construct an example to show that the observability of (A, Q) is not necessary for the solution X of the CARE (13.1.1) to be positive definite.
3. Prove that the matrix

$$M = \begin{pmatrix} A + S(A^{-1})^T Q & -S(A^{-1})^T \\ -(A^{-1})^T Q & (A^{-1})^T \end{pmatrix},$$

$S = BR^{-1}B^T$, associated with the DARE:

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A$$

is symplectic, and that if λ is a nonzero eigenvalue of M , so is $\frac{1}{\lambda}$.

4. Establish the relation (13.2.16).
5. (a) Prove the discrete counterpart of Theorem 13.2.4; that is, prove that the DARE (13.1.2) has a unique symmetric positive semidefinite stabilizing solution if and only if (A, B) is discrete-stabilizable and the associated symplectic matrix M does not have an eigenvalue on the unit circle.
- (b) Prove the discrete counterpart of Theorem 13.2.5; that is prove that if (A, B) is discrete-stabilizable and (A, Q) is discrete-detectable, then the symplectic matrix M defined by (13.3.1) does not have an eigenvalue on the unit circle.
- (c) Using the results of Problem 3, and those of 5(a) and 5(b), prove Theorem 13.3.2.
6. Prove that the homogeneous CARE: $XA + A^T X + XSX = 0$ has a stabilizing solution if A has no eigenvalues on the imaginary axis. Prove or disprove a discrete-counterpart of this result.
7. Prove that the quantity

$$\frac{2 \|A\|_F^2 \frac{\|Q\|_F}{\|X\|_F} + \|A\|_F^2 \|S\|_F \|X\|_F}{sep_d(A_d^T, A_d)},$$

where $A_d = A - B(R + B^T X B)^{-1} B^T X A$, serves as an approximate condition number of the DARE (13.1.2). Construct an example of an ill-conditioned DARE using this quantity.

8. Assume that $A - SX$ is asymptotically stable. Let $H_k, k = 0, 1, 2$ satisfy the Lyapunov equations:

$$(A - SX)^T H_k + H_k (A - SX) = -X^k,$$

Then prove that in 2-norm $\left[\frac{\|H_0\| \|H_2\|}{Cond(X)} \right]^{\frac{1}{2}} \leq \|H_1\| \leq \|H_0\|^{\frac{1}{2}} \|H_1\|^{\frac{1}{2}}$, where X is positive definite.

9. Find an example to illustrate that a small relative residual in a computed solution of the CARE does not guarantee a small error in the solution.
10. Prove that if Ω is singular, then $sep((A - SX), -(A - SX)^T)$ is zero.
11. Give a proof of the theoretical basis of Schur algorithm for the CARE using Theorem 13.2.6. That is, prove the following: Let H be the Hamiltonian matrix associated with the CARE and $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$ be the unitary matrix such that $U^T H U = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$, where the n eigenvalues of H with negative real parts are stacked in T_{11} . Then, $X = U_{21} U_{11}^{-1}$ is the unique positive semidefinite stabilizing solution of the CARE.

12. Construct an example to show that the solution of the CARE, obtained by the Schur method, might be inaccurate, even though the problem is not ill-conditioned. (**Hint:** Construct an example for which U_{11} is ill-conditioned, but the CARE is well-conditioned).
13. Give an example to demonstrate the superiority of the Schur algorithm for the CARE over the eigenvector algorithm, in case the associated Hamiltonian matrix is nearly defective.
14. Using Theorem 13.5.1 and the transformation

$$H = (M + I_{2n})(M - I_{2n})^{-1},$$

prove Theorem 13.5.2.

15. Construct an example to demonstrate the numerical difficulties of the Schur algorithm for the DARE in case the matrix A is nearly singular.
16. Write down an algorithm for solving the discrete algebraic Riccati equation, using the eigenvectors of the symplectic matrix. Discuss the computational drawbacks of the algorithm. Construct an example to illustrate the computational drawbacks.
17. Prove the properties 1 through 5 of the matrix sign-function $\text{Sign}(A)$ stated in Section 13.5.3.
18. Prove that if $|\lambda| = 1$ is an eigenvalue of the pencil $P_{DARE} - \lambda N_{DARE}$ with the eigenvector $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$, where P_{DARE} and N_{DARE} are the same as given in Theorem 13.5.5, then the detectability of (A, Q) implies that $z_1 = 0$.
19. Develop the generalized Schur methods for the CARE and DARE in details.
20. Why is the generalized Schur method not preferable over the Schur method for the CARE if R is not nearly singular?
21. Construct an example to demonstrate the poor accuracy of the generalized eigenvector method for the DARE in case the pencil $P_{DARE} - \lambda N_{DARE}$ has near multiple eigenvalues. Apply the generalized Schur algorithm (Algorithm 13.5.2) to the same example and verify the improvement in the accuracy of the solution.
22. Work out the details of how the pencil $P_{CARE}^E - \lambda N_{CARE}^E$ can be transformed to the compressed pencil $P_{CARE}^{EC} - \lambda N_{CARE}^{EC}$ using the QR factorization of the matrix $\begin{pmatrix} B \\ R \end{pmatrix}$.
23. Repeat the exercise for the DARE; that is, work out the details of the transformation to the pencil $P_{DARE}^{EC} - \lambda N_{DARE}^{EC}$ using the QR factorization of the matrix $\begin{pmatrix} R \\ -B \end{pmatrix}$.

24. Prove that the pencil $P_{CARE}^E - \lambda N_{CARE}^E$ and the pencil $P_{CARE}^{EC} - \lambda N_{CARE}^{EC}$ as defined in Section 13.5.2 for the CARE have the same deflating subspaces; and similarly for the DARE.
25. Develop the following algorithms in detail for both the DCARE and DDARE (consult Laub (1991), Benner (1997)).
- (1) The Schur algorithms
 - (2) The Generalized Schur algorithms
 - (3) The Inverse-free generalized Schur algorithms
 - (4) The Matrix sign-function algorithms
 - (5) Newton's algorithms

Construct a simple example to illustrate each of the above algorithms.

26. Construct an example to demonstrate the superiority of the inverse-free generalized Schur algorithm over the Schur algorithm for the CARE, in case the control weighting matrix R is positive definite but nearly singular.
27. Carry out a numerical experiment with a 150×150 randomly generated problem to make a comparative study with respect to computer-time and accuracy of the solution to the CARE with the following methods: the eigenvector method, the Schur method, inverse-free generalized Schur method, the matrix sign function method and the Hamiltonian structure preserving Schur method. Write down your observations and conclusions.
28. Repeat the previous exercise with the DARE using the following methods: The eigenvector method, the generalized eigenvector method, the Schur method, the generalized Schur method, inverse-free generalized Schur method, and the matrix sign function method.

RESEARCH PROBLEMS ON CHAPTER 13

1. Develop a structure-preserving method to compute the symplectic Schur decomposition and apply the method to solve the DARE; thus obtaining a symplectic structure-preserving method for the DARE.

References

1. J. Abels and P. Benner, CAREX—a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0). *SLICOT Working Note* 1999-14, November 1999a. (Available at the NICONET Website: <http://www.win.tue.ne/niconet/niconet.html>).
2. J. Abels and P. Benner, DAREX—a collection of benchmark examples for discrete-time algebraic Riccati equations (version 2.0). *SLICOT Working Note* 1999-15, November 1999b. (Available at the NICONET Website: <http://www.win.tue.ne/niconet/niconet.html>).
3. G. Ammar and V. Mehrmann, A multishift algorithm for the numerical solution of algebraic Riccati equations, *Electr. Trans. Num. Anal.*, vol. 1, pp. 33-48, 1993.
4. B.D.O. Anderson, and J.B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice Hall, Englewood Cliffs, NJ, 1990.
5. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, 3rd Edition, SIAM, Philadelphia, 1999.
6. T. Ando, *Matrix quadratic equations*, Hokkaido University, Research Institute of Applied Electricity, Division of Applied Mathematics, Sapporo, Japan, 1988.
7. W. Arnold, III and A.J. Laub, Generalized eigenproblem algorithms and software for algebraic Riccati equations, *Proc. IEEE*, vol. 72, pp. 1746-1754, 1984.
8. Z. Bai and J. Demmel, Using the matrix sign function to compute invariant subspaces, *SIAM J. Matrix Anal. Appl.*, vol. 19 , pp. 205-225, 1998.
9. Z. Bai and Q. Qian, Inverse free parallel method for the numerical solution of algebraic Riccati equations, *Proc. Fifth SIAM Conf. Appl. Lin. Alg.*, Snowbird, UT, June, pp. 167-171, 1994, (J. Lewis, Editor).
10. D. Bender and A.J. Laub, The linear-quadratic optimal regulator problem for descriptor systems, *Proc. 24th IEEE Conf. Dec. Control*, Ft. Lauderdale, FLorida, December, pp. 957-962, 1985.
11. D. Bender and A.J. Laub, The linear-quadratic optimal regular for descriptor systems: Discrete-time case, *Automatica*, vol. 23, pp. 71-85, 1987.
12. P. Benner, Computational Methods for Linear-Quadratic Optimization, *Rendiconti del Circulo Matematico di Palermo*, Supplemento, Serie II, no. 58, pp. 21-56, 1999.
13. P. Benner, *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*, Dissertation for Dipl.-Math., Technischen Universität Chemnitz-Zwickau, Germany 1997.

14. P. Benner and R. Byers, An exact line search method for solving generalized continuous-time algebraic Riccati equations, *IEEE Trans. Automat. Control*, pp. 101-107, 1998.
15. P. Benner, R. Byers, V. Mehrmann, and H. Xu, Numerical solution of linear-quadratic control problems for descriptor systems, *Proc. 1999 IEEE Intl. Symp. CACSD*, Kohala Coast-Island of Hawaii, Hawaii, USA, August 22-27, 1999, pp.46-51, 1999, (O. Gonzalez, Editor).
16. P. Benner and H. Fassbender, An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem, *Lin. Alg. Appl.* vol. 263, pp. 75-111, 1997.
17. P. Benner, H. Fassbender, and D. Watkins, Two connections between the SR and HR eigenvalue algorithms, *Lin. Alg. Appl.*, vol. 272, pp.17-32, 1997.
18. P. Benner, H. Fassbender and D. Watkins, SR and SZ algorithms for the symplectic (butterfly) eigenproblem, *Lin. Alg. Appl.*, vol. 287, pp.41-76, 1999.
19. P. Benner, A. Laub, and V. Mehrmann, A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: Continuous-time case, *Tech. Report SPC 95-22, Fak. f. Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, 1995*. (Available at the website: www.math.uni-bremen.de/~benner).
20. P. Benner, A. Laub, and V. Mehrmann, A collection of benchmark examples for the numerical solution of algebraic Riccati equations II: Discrete-time case, *Tech. Report SPC 95-23, Fak. f. Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, 1995*. (Available at the website: www.math.uni-bremen.de/~benner).
21. P. Benner, A. Laub, and V. Mehrmann, Benchmarks for the numerical solution of algebraic Riccati equations. *IEEE Control Systems Magazine*, vol. 7, no. 5, pp. 18-28, 1997.
22. P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga, SLICOT—a subroutine library in systems and control theory. *Applied and Computational Control, Signals, and Circuits*, volume 1, Chapter 10, pp. 499-539. Birkhauser, Boston, MA, 1999. (B.N. Datta, et al., Editors).
23. P. Benner, V. Mehrmann, and H. Xu, A new method for computing the stable invariant subspace of a real Hamiltonian matrix, *J. Comput. Appl. Math.*, vol. 86, pp. 17-43, 1997.
24. P. Benner, V. Mehrmann, and H. Xu, A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils, *Numer. Math.*, vol. 78, pp. 329-358, 1999.
25. G. J. Bierman, Computational aspects of the matrix sign function solution to the ARE, Proc. 23rd *IEEE Conf. Dec. Contr.*, Las Vegas, Nevada, pp. 514-519, 1984.

26. S. Bittanti, A.J Laub, and J.C. Willems, (Editors), *The Riccati Equation*, Springer-Verlag, Berlin, 1991.
27. A. Bunse-Gerstner, V. Mehrmann, and D. Watkins, An SR algorithm for Hamiltonian matrices based on Gaussian elimination, *Methods of Operations Research*, vol. 58, pp. 339-358, 1989.
28. A. Bunse-Gerstner, R. Byers, and V. Mehrmann, A chart of numerical methods for structured eigenvalue problems, *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 2, pp. 419-453, 1992.
29. A. Bunse-Gerstner and V. Mehrmann, A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation, *IEEE Trans. Automat. Control*, AC-31, pp. 1104-1113, 1986.
30. R. Byers, *Hamiltonian and Symplectic Algorithms for the Algebraic Riccati Equation*, PhD thesis, Cornell University, Dept. Comp. Sci., Ithaca, NY, 1983.
31. R. Byers, Numerical condition of the algebraic Riccati equation, *Contemp. Math.*, Amer. Math. Soc., Providence, RI, vol. 47, pp. 35-49, 1985, (R. Brualdi, et al., Editor).
32. R. Byers, A Hamiltonian QR-algorithm, *SIAM J. Sci. Statist. Comput.*, vol. 7, pp. 212-229, 1986a.
33. R. Byers, Numerical stability and instability in matrix sign function based algorithms in *Computational and Combinatorial Methods in Systems Theory*, pp. 185-200, North Holland, New York, 1986b, (C. I. Byrnes and A. Lindquist, Editors).
34. R. Byers, Solving the algebraic Riccati equation with the matrix sign function, *Lin. Alg. Appl.*, vol. 85, pp. 267-279, 1987.
35. R. Byers, A Hamiltonian Jacobi Algorithm, *IEEE Trans. Automat. Control*, vol. 35, no. 5, pp. 566-570, 1990.
36. S. L. Campbell, *Singular Systems of Differential Equations*, Pitman, Marshfield, MA, 1980.
37. C.-H. Chen, Perturbation analysis for solutions of algebraic Riccati equations, *J. Comput. Math.*, vol. 6, pp. 336-347, 1988.
38. W.A. Coppel, Matrix quadratic equations, *Bull. Australian Math. Soc.*, vol. 10, pp. 327-401, 1974.
39. E.D. Denman and A.N. Beavers, The matrix sign function and computations in systems, *Appl. Math. Comput.*, vol. 2, pp. 63-94, 1976.

40. C. E. DeSouza and M. D. Fragoso, On the existence of maximal solution for generalized algebraic Riccati equations arising in stochastic control, *Syst. Contr. Lett.*, vol. 14, pp. 223-239, 1990.
41. C. E. DeSouza, M. R. Gevers, and G. C. Goodwin, Riccati equations in optimal filtering of nonstabilizable systems having singular state transition matrices, *IEEE Trans. Automat. Control*, vol. AC-31, pp. 831-838, 1986.
42. J. Doyle, K. Glover, P. Khargonekar, and B. Francis, State-space solutions to standard H_2 and H_∞ control problems, *IEEE Trans. Automat. Control*, AC-34, pp. 831-847, 1989.
43. A. Emami-Naeini and G. F. Franklin, Design of steady state quadratic loss optimal digital controls for systems with a singular system matrix, *Proc. 13th Asilomar Conference Circ. Syst. Comp.*, pp. 370-374, 1979.
44. A. Emami-Naeini and G. F. Franklin, Deadbeat control and tracking of discrete-time systems, *IEEE Trans. Automat. Control*, vol. AC-27, pp. 176-181, 1982.
45. H. Fassbender and P. Benner, A hybrid method for the numerical solution of discrete-time algebraic Riccati equations, *Contemporary Mathematics on Structured Matrices in Mathematics, Computer Science, and Engineering*, Amer. Math. Soc., Providence, RI, vol. 280, pp. 255-269, 2001, (V. Olshevsky , Editor).
46. H. Fassbender, *Symplectic Method for the Symplectic Eigenproblem*, Kluwer Academic/Plenum Publishers, New York, 2000.
47. W. R. Ferng, W.-W. Lin, and C.-S. Wang, The shift-inverted J -Lanczos algorithm for the numerical solutions of large sparse algebraic Riccati equations, *Comput. Math. Appl.*, vol. 33, no. 10, pp. 23-40, 1997.
48. P.M. Gahinet, A.J. Laub, C.S. Kenney, and G.A. Hewer, *Sensitivity of the stable discrete-time Lyapunov equation*, *IEEE Trans. Automat. Control*, vol. 35, pp. 1209-1217, 1990.
49. J.D. Gardiner, Stabilizing control for second-order models and positive real systems, *AIAA J. Guidance, Dynamics and Control*, vol. 15, pp. 280-282, 1992.
50. J.D. Gardiner and A.J. Laub, A generalization of the matrix-sign-function solution for algebraic Riccati equations, *Int. J. Control*, vol. 44, pp. 823-832, 1986.
51. J.D. Gardiner and A.J. Laub, Parallel algorithms for algebraic Riccati equations, *Int. J. Control*, vol. 54, pp. 1317-1333, 1991.
52. J.D. Gardiner, A.J. Laub, J.J. Amato, and C.B. Moler, Solution of the Sylvester matrix equation $AXB + CXD = E$, *ACM Trans. Math. Software*, vol. 18, pp. 223-231, 1992.

53. A. Ghavimi, C. Kenney, and A.J. Laub, Local convergence analysis of conjugate gradient methods for solving algebraic Riccati equations, *IEEE Trans. Automat. Control*, vol. AC-37, pp. 1062-1067, 1992.
54. A. R. Ghavimi and A.J. Laub, Backward error, sensitivity, and refinement of computed solutions of algebraic Riccati equations, *Num. Lin. Alg. Appl.*, vol. 2, pp. 29-49, 1995.
55. T. Gudmundsson, C. Kenney, and A.J. Laub, Scaling of the discrete-time algebraic Riccati equation to enhance stability of the Schur method, *IEEE Trans. Automat. Control*, AC-37, pp. 513-518, 1992.
56. C.-H. Guo, Newton's method for discrete algebraic Riccati equations when the closed-loop matrix has eigenvalues on the unit circle, *SIAM J. Matrix Anal. Appl.*, vol. 20, pp. 279-294, 1998.
57. C.-H. Guo and P. Lancaster, Analysis and modification of Newton's method for algebraic Riccati equations. *Math. Comp.*, vol. 67, pp. 1089-1105, 1998.
58. C.-H. Guo and A. J. Laub, on a Newton-like method for solving algebraic Riccati equations, *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 694-698, 2000.
59. S. J. Hammarling, Newton's method for solving the algebraic Riccati equation, NPL Report DITC 12/82, National Physical Laboratory, Teddington, Middlesex TW11 OLW, U.K., 1982.
60. G. A. Hewer, An iterative technique for the computation of steady state gains for the discrete optimal controller, *IEEE Trans. Automat. Control*, vol. AC-16, pp. 382-384, 1971.
61. B. Hinrichsen, B. Kelb, and A. Linnemann, An algorithm for the computation of the structured complex stability radius, *Automatica*, vol. 25, pp. 771-775, 1989.
62. D. Hinrichsen and A.J. Pritchard, Stability radii of linear systems, *Sys. & Contr. Lett.*, vol. 7, pp. 1-10, 1986.
63. D. Hinrichsen, A.J. Pritchard, and S.B. Townley, Riccati equation approach to maximizing the complex stability radius by state feedback, *Int. J. Control*, vol. 52, pp. 769-794, 1990.
64. T. Ho, *A study of computational methods for the continuous-time algebraic Riccati equation*, M.Sc. Thesis, Northern Illinois University, DeKalb, Illinois, 2000.
65. J. L. Howland, The sign matrix and the separation of matrix eigenvalues, *Lin. Alg. Appl.*, vol. 49, pp. 221-232, 1983.
66. V. Ionescu, C. Oara, and M. Weiss, *Generalized Riccati Theory and Robust Control*, John Wiley, New York, 1999.

67. I. M. Jaimoukha and E. M. Kasenally, Krylov subspace methods for solving large Lyapunov equations, *SIAM J. Numer. Anal.* vol. 31, 227-251, 1994.
68. M. Jamshidi, An overview on the solutions of the algebraic Riccati equation and related problems, *Large-Scale Systems*, vol. 1, pp. 167-192, 1980.
69. C.S. Kenney and G. Hewer, The sensitivity of the algebraic and differential Riccati equations, *SIAM J. Contr. Optimiz.*, vol. 28, pp. 50-69, 1990.
70. C.S. Kenney and A.J. Laub, On scaling Newton's method for polar decomposition and the matrix sign function, *Proc. 1990 Amer. Control Conf.*, pp. 2560-2564, 1990, and *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 688-706, 1992.
71. C.S. Kenney and A.J. Laub, The matrix sign function, *IEEE Trans. Automat. Control*, vol. 40, pp. 1330-1348, 1995.
72. 74C.S. Kenney, A.J. Laub, and M. Wette, A stability-enhancing scaling procedure for Schur-Riccati solvers, *Sys. Contr. Lett.*, vol. 12, pp. 241-250, 1989.
73. C.S. Kenney, A.J. Laub, and M. Wette, Error bounds for Newton refinement of solutions to algebraic Riccati equations, *Math. Control, Signals, and Systems*, vol. 3, pp. 211-224, 1990.
74. H. Kimura, *Chain-scattering Approach to H^∞ -control*, Birkhäuser, Boston, 1996.
75. D. L. Kleinman, On an iterative technique for Riccati equation computations, *IEEE Trans. Automat. Control*, AC-13, pp. 114-115, 1968.
76. D.L. Kleinman, Stabilizing a discrete, constant linear system with application to iterative methods for solving the Riccati equation, *IEEE Trans. Automat. Control*, vol. AC-19, pp. 252-254, 1974.
77. M.M. Konstantinov, P.H. Petkov, and N.D. Christov, Perturbation analysis of matrix quadratic equations, *SIAM J. Sci. Stat. Comput.*, vol. 11, pp. 1159-1163, 1990.
78. M.M. Konstantinov, P.H. Petkov, and N.D. Christov, Perturbation analysis of the discrete Riccati equation, *Kybernetika*, vol. 29, pp. 18-29, 1993.
79. M.M. Konstantinov, P.H. Petkov, D.W. Gu, and I. Postlethwaite, *Perturbation Techniques for Linear Control Problems*, Report 95-7, Control Systems Research, Department of Engineering, Leicester University, UK, 1995.
80. V. Kučera, A contribution to matrix quadratic equations, *IEEE Trans. Automat. Control*, vol. 17, pp. 344-347, 1972.
81. V. Kučera, *Discrete linear control*, John Wiley & Sons, New York, 1979.

82. H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley Interscience, New York, 1972.
83. P. Lancaster and L. Rodman, Existence and uniqueness theorems for algebraic Riccati equations, *Int. J. Control.*, vol. 32, pp. 285-309, 1980.
84. P. Lancaster and L. Rodman, *The Algebraic Riccati Equations*, Oxford University Press, Oxford, 1995.
85. A. Laub, A Schur method for solving algebraic Riccati equations, *IEEE Trans. Automat. Control*, AC-24, pp. 913-921, 1979.
86. A. Laub, *Invariant subspace methods for the numerical solution of Riccati equations*, in *The Riccati Equation*, pp. 163-196, 1991. (S. Bittanti, et al., Editors).
87. W.-W. Lin, A new method for computing the closed-loop eigenvalues of a discrete-time algebraic Riccati equation, *Lin. Alg. Appl.*, vol. 6, pp. 157-180, 1987.
88. W.-W. Lin, and T.-C. Ho, *On Schur type decompositions of Hamiltonian and Symplectic Pencils*, Tech. Report, Institute of Applied Mathematics, National Tsing Hua University Taiwan, 1990.
89. L. Lu, and W.-W. Lin, An iterative algorithm for the solution of the discrete-time algebraic Riccati equation, *Lin. Alg. Appl.*, vol. 188/189, pp. 465-488, 1993.
90. A. McFarlane, An eigenvector solution of the optimal linear regulator problem, *J. Electronics Control*, vol. 14, pp. 643-654, 1963.
91. V.L. Mehrmann, *The Autonomous Linear Quadratic Control Problem*. Lecture Notes in Control and Information Sciences, vol. 163, Springer Verlag, Berlin, 1991.
92. V. Mehrmann, A symplectic orthogonal method for single-input single-output discrete-time optimal linear quadratic control problems, *SIAM J. Matrix Anal. Appl.*, pp. 221-248, 1988.
93. C. Paige and C. Van Loan, A Schur decomposition for Hamiltonian matrices, *Lin. Alg. Appl.*, pp. 11-32, 1981.
94. P. Pandey, On scaling an algebraic Riccati equation, in *Proc. Amer. Control Conference*, San Francisco, CA, pp. 1583-1587, 1993.
95. T. Pappas, A. Laub, and N. Sandell, On the numerical solution of the discrete-time algebraic Riccati equation, *IEEE Trans. Automat. Control*, AC-25, pp. 631-641, 1980.
96. R. V. Patel, On computing the eigenvalues of a symplectic pencil, *Lin. Alg. Appl.* vol. 188/189, pp. 591-611, 1993.

97. R. V. Patel, Z. Lin, and P. Misra, Computation of stable invariant subspaces of Hamiltonian matrices, *SIAM J. Matrix Anal. Appl.*, vol. 15, pp. 284-298, 1994.
98. I.R. Petersen, Disturbance attenuation and H^∞ -optimization: A design method based on the algebraic Riccati equations, *IEEE Trans. Autom. Control*, vol. 32, pp. 427-429, 1987.
99. P. Petkov, N. Christov and M. Konstantinov, Numerical properties of the generalized Schur approach for solving the discrete matrix Riccati equation, *Proc. 18th Spring Conference of the Union of Bulgarian Mathematicians*, Albena, pp. 452-457, 1989.
100. P. Petkov, N. Christov and M. Konstantinov, On the numerical properties of the Schur approach for solving the matrix Riccati equation, *Syst. Contr. Lett.*, vol. 9, pp. 197-201, 1987.
101. P. Petkov, N. Christov and M. Konstantinov, *Computational Methods for Linear Control Systems*, Prentice Hall, UK, 1991.
102. P. Petkov, M. Konstantinov, D. Gu and I. Postlethwaite, *Solving continuous-time matrix algebraic Riccati equations with condition and accuracy estimates*, Tech. Report 94-64, Control Systems Research, Department of Engineering, Leicester University, Leicester LE1 7RH, UK, Dec. 1994.
103. P. Petkov, M.M. Konstantinov, and V. Mehrmann, DGRSVX and DMSRIC: *Fortran 77 subroutines for solving continuous-time matrix algebraic Riccati equations with condition and accuracy estimates*. Technical Report SFB393/98-116, Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz, FRG, 1998.
104. J.E. Potter, Matrix quadratic solutions, *SIAM J. Appl. Math.*, vol. 14, pp. 496-501, 1966.
105. E. Quintana and V. Hernández, *Algoritmos por bloques y paralelos para resolver ecuaciones matriciales de Riccati mediante el método de Newton*, Tech. Report DSIC-II/6/96, Dpto. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain, 1996a.
106. E. Quintana and V. Hernández, *Algoritmos por bloques y paralelos para resolver ecuaciones matriciales de Riccati mediante el método de Schur*, Tech. Report DSIC-II/7/96, Dpto. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain, 1996b.
107. E. Quintana and V. Hernández, *Algoritmos por bloques y paralelos para resolver ecuaciones matriciales de Riccati mediante la división espectral*, Tech. Report DSIC-II/6/96, Dpto de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain, 1996c.

108. J. Roberts, Linear model reduction and solution of the algebraic Riccati equation by use of the sign function, *Int. J. Control.*, vol. 32, pp. 677-687, 1980 (reprint of a technical report from Cambridge University in 1971).
109. N. Sandell, On Newton's method for Riccati equation solution, *IEEE Trans. Auto. Control*, vol. AC-19, pp. 254-255, 1974.
110. V. Sima, Algorithms for linear-quadratic optimization, Vol. 200, *Pure and Applied Mathematics*, Marcel Dekker, New York, 1996.
111. V. Sima, An efficient Schur method to solve the stabilizing problem, *IEEE Trans. Automat. Control*, AC-26, pp. 724-725, 1981.
112. V. Sima, P. Petkov and S. Van Huffel, Efficient and reliable algorithms for condition estimation of Lyapunov and Riccati equations, *Proc. 14th Int. Symposium of Mathematical Theory of Networks and Systems* (MTNS - 2000), Perpignan, France, June 19-23, 2000.
113. M. A. Singer and S. J. Hammarling, The Algebraic Riccati Equation, *National Physical Laboratory Report*, DITC 23/83, January, 1983.
114. G.W. Stewart, Algorithm 506-HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix, *ACM Trans. Math. Software*, vol. 2, pp. 275-280, 1976.
115. J.-G. Sun, Residual bounds of approximate solutions of the algebraic Riccati equations, *Numer. Math.*, vol. 76, pp. 249-263, 1997a.
116. J.-G. Sun, Backward error for the discrete-time algebraic Riccati equation, *Lin. Alg. Appl.*, vol. 25, pp. 183-208, 1997b.
117. J.-G. Sun, Perturbation analysis of the matrix sign function, *Lin. Alg. Appl.*, vol. 250, pp. 177-206, 1997c.
118. J.-G. Sun, Perturbation theory for algebraic Riccati equation. *SIAM J. Matrix Anal. Appl.*, vol. 19, no. 1, pp. 39-65, 1998.
119. P. Van Dooren, A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Sci. Stat. Comput.*, vol. 2, pp. 121-135, 1981.
120. P. Van Dooren, Algorithm 590-DSUBSP and EXCHQZ: Fortran subroutines for computing deflating subspaces with specified spectrum, *ACM Trans. Math. Software*, vol. 8, pp. 376-382, 1982.
121. C. F. Van Loan, A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix, *Lin. Alg. Appl.*, vol. 16, pp. 233-251, 1984.

- 122. J.C. Willems, Least squares stationary optimal control and the algebraic Riccati equation. *IEEE Trans. Autom. Control*, vol. AC-16, pp. 621-634, 1971.
- 123. H.K. Wimmer, The algebraic Riccati equation: Conditions for the existence and uniqueness of solutions. *Lin. Alg. Appl.*, vol. 58, pp. 441-452, 1984.
- 124. H.K. Wimmer, Existence of positive-definite and semidefinite solutions of discrete-time algebraic Riccati equations. *Int. J. Control*, vol. 59, pp. 463-471, 1994.
- 125. S.-F. Xu, Sensitivity analysis of the algebraic Riccati equations, *Numer. Math.*, vol. 75, pp. 121-134, 1996.
- 126. G. Zames, Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *IEEE Trans. Automat. Control*, vol. 26, pp. 301-320, 1981.
- 127. L.Q. Zhang, J. Lam, and Q.L. Zhang, Lyapunov and Riccati equations of discrete-time descriptor systems, *IEEE Trans. Automat. Control*, vol. 44, no. 1, pp. 2134-2139, 1999.
- 128. K. Zhou, K. Glover, B. Bodenheimer, and J. Doyle, Mixed H_2 and \mathcal{H}_∞ performance objectives I: Robust performance analysis; *IEEE Trans. Automat. Control*, vol. 39, pp. 1564-1574, 1994.
- 129. K. Zhou and P. Khargonekar, An algebraic Riccati equation approach to H_∞ optimization, *Syst. Contr. Lett.*, vol. 11, pp. 317-319, 1988.
- 130. 132 K. Zhou, J. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice Hall, Upper Saddle River, NJ, 1996.

Chapter 14

INTERNAL BALANCING AND MODEL REDUCTION

Contents

14.1 Introduction	758
14.2 Internal Balancing for Continuous-time Systems	759
14.2.1 Internal Balancing of a Minimal Realization	759
14.2.2 Internal Balancing of a Nonminimal Realization	764
14.3 Internal Balancing For Discrete-time Systems	766
14.4 Model Reduction	768
14.4.1 Model Reduction via Balanced Truncation	769
14.4.2 The Schur Method for Model Reduction	772
14.4.3 A Balancing-Free Square-Root Method for Model Reduction	780
14.5 Hankel-Norm Approximations	781
14.5.1 A Characterization of All Solutions to Hankel-Norm Approximation .	781
14.6 Model Reduction of an Unstable System	790
14.7 Frequency Weighted Model Reduction	791
14.8 Summary and Comparisons of Model Reduction Procedures	792
14.9 Some Selected Software	794
14.9.1 MATLAB CONTROL SYSTEM TOOLBOX	794
14.9.2 MATCONTROL	794
14.9.3 CSP-ANM	794
14.9.4 SLICOT	794
14.9.5 MATRIX _X	794
14.10 Summary and Review	795
14.11 Chapter Notes and Further Reading	797

Topics Covered

- Internal Balancing
- Model Reduction via Internal Balancing
- Model Reduction via Schur Decomposition
- Hankel Norm Approximation

14.1 Introduction

Several practical situations such as the design of large space structures (LSS), control of power systems, and others, give rise to very large-scale control problems. Typically, these come from the discretization of distributed parameter problems and have thousands of states in practice. Enormous computational complexities hinder the computationally feasible solutions of these problems.

As a result, control theorists have always sought ways to construct *reduced-order models* of appropriate dimensions (depending upon the problem to be solved) which can then be used in practice in the design of control systems. This process is known as *model reduction*. The idea of model reduction is to construct a reduced-order model from the original full-order model such that the reduced-order model is close, in some sense, to the full-order model. The closeness is normally measured by the smallness of $\| G(s) - G_R(s) \|$ where $G(s)$ and $G_R(s)$ are, respectively, the transfer function matrices of the original and the reduced-order model. Two norms, $\| \cdot \|_\infty$ norm and the Hankel-norm have been considered here. The problem of constructing a reduced order model such that the error is minimized using Hankel-norm is called an **Optimal Hankel-norm approximation**. A widely used practice of model reduction is to first find a balanced realization (that is, a realization with controllability and observability Grammians equal to a diagonal matrix) and then to truncate the balanced realization in an appropriate manner to obtain a reduced-order model. The process is known as **balanced truncation method**. Balancing of a continuous-time system is discussed in **Section 14.2**, where two algorithms are described. **Algorithm 14.2.1** (Laub (1980), Glover (1984)) constructs internal balancing of a stable, controllable and observable system, whereas **Algorithm 14.2.2** (Tombs and Postlethwaite (1987)) is applicable to any stable system. **Internal balancing** of a discrete-time system is described in **Section 14.3**.

In **Section 14.4** it is shown (**Theorem 14.4.1**) that a reduced-order model constructed by **truncating a balanced realization remains stable** and the H_∞ -norm error is bounded.

A Schur method (**Algorithm 14.4.2**) for model reduction is then described. The Schur method due to Safanov and Chiang (1989) is designed to overcome the numerical difficulties in **Algorithm 14.2.1** due to the possible ill-conditioning of the transforming matrices. In **Theorem**

14.4.2, it is shown that the transfer function matrix of the reduced-order model obtained by the Schur method is the same as that of the model reduction procedure via internal balancing using Algorithm 14.2.1. The Schur method, however, has its own computational problem. It requires computation of the product of the controllability and observability Grammians, which might be a source of round-off errors. The method, however can be modified by using Cholesky factors of the Grammians which then leads to the **square-root algorithm (Algorithm 14.2.2)**. The advantages of the Schur and the square-root methods can be combined into a **balancing-free square-root algorithm** (Varga (1991)). This algorithm is briefly sketched in **Section 14.4.3**.

Section 14.5 is concerned with **Hankel-norm approximation**. A state-space characterization of all solutions to optimal Hankel-norm approximation due to Glover (1984) is stated (**Theorem 14.5.2**) and then an algorithm to compute an **optimal Hankel-norm approximation (Algorithm 14.5.1)** is described.

Section 14.6 shows how to obtain a **reduced-order model of an unstable system**.

The **frequency weighted model reduction** problem due to Enns (1984) is considered in **Section 14.7**. The errors at the high frequencies can sometimes possibly be reduced by using suitable weights on the frequencies.

Finally, in **Section 14.8**, a numerical **comparison of different model reduction procedures** is given.

14.2 Internal Balancing for Continuous-time Systems

Let (A, B, C) be an n -th order realization of a **stable** continuous-or discrete-time system. Then it is known (Glover (1984)) that there exists a transformation such that controllability and observability Grammians for the transformed realization are equal to a diagonal matrix Σ . Such a realization is called a **balanced realization (or internally balanced realization)**. Internal balancing of a given realization is a preliminary step to a class of methods for model reduction, called **Balance Truncation Methods**. In this section, we describe two algorithms for internal balancing of a **continuous-time system**. The first algorithm (Laub (1980), Glover (1984)) requires that the given realization is minimal; that is, the system is both controllable and observable. The second one (Tombs and Postlethwaite (1987)) is designed for a nonminimal realization. The matrix D of the system (A, B, C, D) remains unchanged during the transformation of the system to a balanced system. **We, therefore, drop the matrix D from our discussions in this chapter.**

14.2.1 Internal Balancing of a Minimal Realization

Assume that the realization of the system (A, B, C, D) is minimal. Thus, it is both controllable and observable. Therefore, the controllability Grammian C_G and the observability Grammian O_G are symmetric and positive definite (see **Chapter 7**) and hence admit the Cholesky factorizations.

Let $C_G = L_c L_c^T$ and $O_G = L_o L_o^T$ be the respective Cholesky factorizations.

Let

$$L_o^T L_c = U \Sigma V^T \quad (14.2.1)$$

be the singular value decomposition of $L_o^T L_c$.

Define now

$$T = L_c V \Sigma^{-\frac{1}{2}}, \quad (14.2.2)$$

where $\Sigma^{\frac{1}{2}}$ denotes the square root of Σ .

Then T is nonsingular, and furthermore using the expressions for C_G and the equation (14.2.2), we see that the transformed controllability Grammian \tilde{C}_G is

$$\tilde{C}_G = T^{-1} C_G T^{-T} = \Sigma^{\frac{1}{2}} V^T L_c^{-1} L_c L_c^T L_c^{-T} V \Sigma^{\frac{1}{2}} = \Sigma.$$

Similarly, using the expression for O_G and the equations (14.2.1) and (14.2.2), we see that the transformed observability Grammian \tilde{O}_G is

$$\tilde{O}_G = T^T O_G T = \Sigma^{-\frac{1}{2}} V^T L_c^T L_o L_o^T L_c V \Sigma^{-\frac{1}{2}} = \Sigma^{-\frac{1}{2}} V^T V \Sigma U^T U \Sigma V^T V \Sigma^{-\frac{1}{2}} = \Sigma^{\frac{1}{2}} \cdot \Sigma^{\frac{1}{2}} = \Sigma.$$

Thus the particular choice of

$$T = L_c V \Sigma^{-\frac{1}{2}} \quad (14.2.3)$$

reduces both the controllability and observability Grammians to the same diagonal matrix Σ . The system $(\tilde{A}, \tilde{B}, \tilde{C})$, where the system matrices are defined by

$$\tilde{A} = T^{-1} A T, \quad \tilde{B} = T^{-1} B, \quad \tilde{C} = C T \quad (14.2.4)$$

is then a **balanced realization** of the system (A, B, C) . The decreasing positive numbers $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$ in $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$, are the **Hankel singular values**.

The above discussion leads to the following algorithm for internal balancing.

Algorithm 14.2.1 An Algorithm for Internal Balancing of a Continuous-time Minimal Realization

Inputs:

A – The $n \times n$ state matrix

B – The $n \times m$ input matrix

C – The $r \times n$ output matrix

Outputs:

T – An $n \times n$ nonsingular balancing transforming matrix
 $\tilde{A}, \tilde{B}, \tilde{C}$ – The matrices of internally balanced realization:

$$\tilde{A} = T^{-1}AT, \tilde{B} = T^{-1}B, \tilde{C} = CT.$$

Assumptions: (A, B) is controllable, (A, C) is observable and A is stable.

Result:

$T^{-1}C_G T^{-T} = T^T O_G T = \Sigma$, a diagonal matrix with positive diagonal entries.

Step 1. Compute the controllability and observability Grammians, C_G and O_G , by solving, respectively, the Lyapunov equations:

$$AC_G + C_G A^T + BB^T = 0 \quad (14.2.5)$$

$$A^T O_G + O_G A + C^T C = 0 \quad (14.2.6)$$

(Note that since A is a stable matrix, the matrices C_G and O_G can be obtained by solving the respective Lyapunov equations above (see **Chapter 7**)).

Step 2. Find the Cholesky factors L_c and L_o of C_G and O_G :

$$\begin{aligned} C_G &= L_c L_c^T \\ O_G &= L_o L_o^T \end{aligned} \quad (14.2.7)$$

Step 3. Find the singular value decomposition (SVD) of the matrix $L_o^T L_c$:

$$L_o^T L_c = U \Sigma V^T.$$

Step 4. Compute $\Sigma^{-\frac{1}{2}} = \text{diag}\left(\frac{1}{\sqrt{\sigma_1}}, \frac{1}{\sqrt{\sigma_2}}, \dots, \frac{1}{\sqrt{\sigma_n}}\right)$, where $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$. (Note that $\sigma_i, i = 1, 2, \dots, n$ are positive).

Step 5. Form $T = L_c V \Sigma^{-\frac{1}{2}}$

Step 6. Compute the matrices of the balanced realization:

$$\begin{aligned} \tilde{A} &= T^{-1}AT \\ \tilde{B} &= T^{-1}B \\ \tilde{C} &= CT \end{aligned}$$

Remark. The original method of Laub (1980) consisted in finding the transforming matrix T by diagonalizing the product $L_c^T O_G L_c$ or $L_o^T C_G L_o$, which is symmetric and positive definite. The method described here is mathematically equivalent to Laub's method and is numerically more effective.

Example 14.2.1 Consider finding the balanced realization using Algorithm 14.2.1 of the system (A, B, C) given by:

$$A = \begin{pmatrix} -1 & 2 & 3 \\ 0 & -2 & 1 \\ 0 & 0 & -3 \end{pmatrix}, \quad B = (1, 1, 1)^T, \quad C = (1, 1, 1).$$

Step 1. By solving the Lyapunov equation (14.2.5), we obtain

$$C_G = \begin{pmatrix} 3.9250 & 0.9750 & 0.4917 \\ 0.9750 & 0.3667 & 0.2333 \\ 0.4917 & 0.2333 & 0.1667 \end{pmatrix}.$$

Similarly, by solving the Lyapunov equation (14.2.6), we obtain

$$O_G = \begin{pmatrix} 0.5000 & 0.6667 & 0.7917 \\ 0.6667 & 0.9167 & 1.1000 \\ 0.7917 & 1.1000 & 1.3250 \end{pmatrix}.$$

Step 2. The Cholesky factors of C_G and O_G are:

$$L_c = \begin{pmatrix} 1.9812 & 0 & 0 \\ 0.4912 & 0.3528 & 0 \\ 0.2482 & 0.3152 & 0.0757 \end{pmatrix}, \quad L_o = \begin{pmatrix} 0.7071 & 0 & 0 \\ 0.9428 & 0.1667 & 0 \\ 1.1196 & 0.2667 & 0.0204 \end{pmatrix}.$$

Step 3. From the singular value decomposition of $L_o^T L_c$ (using MATLAB function *svd*):

$$[U, \Sigma, V] = \text{svd}(L_o^T L_c)$$

we have

$$\Sigma = \text{diag} \left(\begin{array}{ccc} 2.2589 & 0.0917 & 0.0006 \end{array} \right)$$

$$V = \begin{pmatrix} 0.9507 & -0.3099 & 0.0085 \\ 0.3076 & 0.9398 & -0.1488 \\ 0.0381 & 0.1441 & 0.9888 \end{pmatrix}.$$

Step 4.

$$\Sigma^{\frac{1}{2}} = \text{diag}(1.5030, 0.3028, 0.0248).$$

Step 5. The transforming matrix T is:

$$T = L_c V \Sigma^{-\frac{1}{2}} = \begin{pmatrix} 1.2532 & -2.0277 & 0.6775 \\ 0.3835 & 0.5914 & -1.9487 \\ 0.2234 & 0.7604 & 1.2131 \end{pmatrix}.$$

Step 6. The balanced matrices are:

$$\tilde{A} = T^{-1}AT = \begin{pmatrix} -0.7659 & 0.5801 & 0.0478 \\ -0.5801 & -2.4919 & -0.4253 \\ -0.0478 & -0.4253 & -2.7422 \end{pmatrix}.$$

$$\tilde{B} = T^{-1}B = \begin{pmatrix} 1.8602 \\ 0.6759 \\ 0.0581 \end{pmatrix}.$$

$$\tilde{C} = CT = \begin{pmatrix} 1.8602 & -0.6759 & -0.0581 \end{pmatrix}.$$

Verify:

$$\begin{aligned} T^{-1}C_G T^{-T} &= T^T O_G T = \Sigma \\ &= \text{diag}(2.2589, 0.0917, 0.0006). \end{aligned}$$

Computational Remarks: The explicit computation of the product $L_o^T L_c$ can be a source of round-off errors. The small singular values might be almost wiped out by the rounding errors in forming the explicit product $L_o^T L_c$. It is suggested that the algorithm of Heath et al. (1986), that computes the singular values of a product of two matrices without explicitly forming the product, be used in practical implementation of this algorithm.

MATLAB NOTES: The MATLAB function in the form:

$$[\tilde{A}, \tilde{B}, \tilde{C}] = \text{balreal}(A, B, C)$$

returns the balanced realization of the system (A, B, C) . The use of the function **balreal** in the following format:

$$[\tilde{A}, \tilde{B}, \tilde{C}, G, T] = \text{balreal}(A, B, C)$$

returns, in addition to $\tilde{A}, \tilde{B}, \tilde{C}$, a vector G containing the diagonal of the Grammian of the balanced realization, and the matrix T , the matrix of the similarity transformation that transforms (A, B, C) to $(\tilde{A}, \tilde{B}, \tilde{C})$.

MATCONTROL NOTES: Algorithm 14.2.1 has been implemented in MATCONTROL function **balsvd**.

14.2.2 Internal Balancing of a Nonminimal Realization

In the previous section we showed how to compute the balanced realization of a stable minimal realization. Now we show how to obtain a balanced realization given a stable **nonminimal continuous-time realization**. The method is due to Tombs and Postlethwaite (1987) and is known as the **square-root method**. The algorithm is based on a partitioning of the singular value decomposition of the product $L_o^T L_c$ and the balanced matrices $\tilde{A}, \tilde{B}, \tilde{C}$ are found by applying two transforming matrices L and Z to the matrices A, B , and C . The matrices L_o and L_c are, respectively, the Cholesky factors of the positive semidefinite observability and controllability matrices O_G and C_G .

The balanced realization in this case is of order k ($k < n$) in contrast with the previous one where the balanced matrices are of the same orders as of the original model.

Let the SVD of $L_o^T L_c$ be represented as

$$L_o^T L_c = (U_1, U_2) \text{diag}(\Sigma_1, \Sigma_2) (V_1, V_2)^T$$

where $\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_k) > 0$, $\Sigma_2 = 0_{n-k \times n-k}$.

The matrices U_1, V_1^T and Σ_1 are of order $n \times k$, $k \times n$ and $k \times k$, respectively.

Define now

$$L = L_o U_1 \Sigma_1^{-\frac{1}{2}}, \quad Z = L_c V_1 \Sigma_1^{-\frac{1}{2}}$$

Then it has been shown in Tombs and Postlethwaite (1987) that the realization $(\tilde{A}, \tilde{B}, \tilde{C})$, where the matrices \tilde{A}, \tilde{B} and \tilde{C} are defined by $\tilde{A} = L^T A Z$, $\tilde{B} = L^T B$ and $\tilde{C} = C Z$ is minimal and balanced, truncated to k states of the system (A, B, C) .

Remark: Note that no assumption on the controllability of (A, B) or the observability of (A, C) is made.

Algorithm 14.2.2 The Square-Root Algorithm for Balanced Realization of a Continuous-time Nonminimal Realization

Inputs: The system matrices A, B and C of a nonminimal realization

Outputs: The transforming matrices L, Z and the balanced matrices \tilde{A}, \tilde{B} and \tilde{C} .

Assumption: A is stable.

Step 1. Compute L_o and L_c , using the LDL^T decomposition of O_G and C_G , respectively. (Note that L_o and L_c may be symmetric positive semidefinite, rather than positive definite).

Step 2. Compute the SVD of $L_o^T L_c$ and partition it in the form:

$$L_o^T L_c = (U_1, U_2) \text{diag}(\Sigma_1, \Sigma_2) (V_1, V_2)^T$$

where $\Sigma_1 = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_k) > 0$.

Step 3. Define

$$L = L_o U_1 \Sigma_1^{-\frac{1}{2}}$$

and

$$Z = L_c V_1 \Sigma_1^{-\frac{1}{2}}.$$

Step 4. Compute the balanced matrices \tilde{A} , \tilde{B} and \tilde{C} as:

$$\tilde{A} = L^T A Z, \quad \tilde{B} = L^T B \text{ and } \tilde{C} = C Z.$$

Example 14.2.2 Let A, C be the same as in Example 14.2.1, and let $B = (1, 0, 0)^T$. Thus (A, B) is not controllable.

Step 1.

$$L_o = \begin{pmatrix} 0.7071 & 0 & 0 \\ 0.9428 & 0.1667 & 0 \\ 1.1196 & 0.2667 & 0.0204 \end{pmatrix}, \quad L_c = \text{diag}(0.7071, 0, 0).$$

$$\text{Step 2. } U_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad V_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \Sigma_1 = 0.5000. \quad k = 1.$$

Step 3.

$$L = L_o U_1 \Sigma_1^{-\frac{1}{2}} = \begin{pmatrix} 1 \\ 1.3333 \\ 1.5833 \end{pmatrix}$$

$$Z = L_c V_1 \Sigma_1^{-\frac{1}{2}} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Step 4. $\tilde{A} = -1, \tilde{B} = 1, \tilde{C} = 1$.

Thus, $(\tilde{A}, \tilde{B}, \tilde{C})$ is a **balanced realization** of order 1; since the realized system is both controllable and observable. Indeed both the controllability and observability Grammians for this realization are equal to 0.5.

MATCONTROL Note. Algorithm 14.2.2 has been implemented in MATCONTROL function **balsqt**.

Numerical Difficulties of Algorithm 14.2.1 and 14.2.2.

Algorithm 14.2.1 of the last section can be numerically unstable in the case when the matrix T is ill-conditioned.

To see this, we borrow the following simple example from Safonov and Chiang (1989):

$$\text{Let } \left(\begin{array}{c|c} A & B \\ \hline C & 0 \end{array} \right) = \left(\begin{array}{cc|c} -\frac{1}{2} & -\epsilon & \epsilon \\ 0 & -\frac{1}{2} & 1 \\ \hline 1 & \epsilon & 0 \end{array} \right)$$

on the The transforming matrix T of Algorithm 14.2.1 in this case is given by

$$T = \begin{pmatrix} \sqrt{\frac{1}{\epsilon}} & 0 \\ 0 & \sqrt{\epsilon} \end{pmatrix}.$$

Thus as ϵ becomes smaller and smaller, T becomes more and more ill-conditioned. Indeed, when $\epsilon \rightarrow 0$, $\text{Cond}(T)$ becomes infinite.

In such cases, the model reduction procedure via internal balancing becomes unstable.

Similarly, the square-root algorithm (Algorithm 14.2.2) can be unstable if the matrices L and T are ill-conditioned.

14.3 Internal Balancing For Discrete-time Systems

In this section, we consider internal balancing of the stable discrete-time system:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k. \end{aligned} \tag{14.3.1}$$

We assume that the system is controllable, and observable, and give here a discrete analogue of Algorithm 14.2.1.

The discrete analogue of Algorithm 14.2.2 can be similarly developed and is left as an [Exercise 11c].

The controllability Grammian C_G^D and the observability Grammian O_G^D , defined by:

$$C_G^D = \sum_{i=0}^{\infty} A^i B B^T (A^T)^i$$

and

$$O_G^D = \sum_{i=0}^{\infty} (A^T)^i C^T C A^i$$

satisfy, in this case, respectively, the discrete-Lyapunov equations:

$$AC_G^D A^T - C_G^D + BB^T = 0 \tag{14.3.2}$$

and

$$A^T O_G^D A - O_G^D + C^T C = 0. \tag{14.3.3}$$

It can then be shown that the transforming matrix T defined by

$$T = L_c V \Sigma^{-\frac{1}{2}}, \tag{14.3.4}$$

where L_c , V , and Σ are defined in the same way as in the continuous-time case, will transform the system (14.3.1) to the internally balanced system:

$$\begin{aligned}\tilde{x}_{k+1} &= \tilde{A}\tilde{x}_k + \tilde{B}u_k, \\ \tilde{y}_k &= \tilde{C}\tilde{x}_k.\end{aligned}\tag{14.3.5}$$

The Grammians again are transformed to the same diagonal matrix Σ , the matrices \tilde{A}, \tilde{B} , and \tilde{C} are defined in the same way as in the continuous-time case.

Example 14.3.1

$$A = \begin{pmatrix} 0.0010 & 1 & 1 \\ 0 & 0.1200 & 1 \\ 0 & 0 & -0.1000 \end{pmatrix},$$

$$B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad C = (1, 1, 1).$$

(Note that the eigenvalues of A have modulii less than 1, so it is discrete-stable).

Step 1. The discrete controllability and observability Grammians obtained, respectively, by solving (14.3.2) and (14.3.3) are:

$$C_G^D = \begin{pmatrix} 6.0507 & 3.2769 & 0.8101 \\ 3.2769 & 2.2558 & 0.8883 \\ 0.8101 & 0.8883 & 1.0101 \end{pmatrix},$$

$$O_G^D = \begin{pmatrix} 1 & 1.0011 & 1.0019 \\ 1.0011 & 2.2730 & 3.2548 \\ 1.0019 & 3.2548 & 5.4787 \end{pmatrix}.$$

Step 2. The Cholesky factors of C_G^D and O_G^D are:

$$L_c^D = \begin{pmatrix} 2.4598 & 0 & 0 \\ 1.3322 & 0.6936 & 0 \\ 0.3293 & 0.6482 & 0.6939 \end{pmatrix}$$

$$L_o^D = \begin{pmatrix} 1 & 0 & 0 \\ 1.0011 & 1.1273 & 0 \\ 1.0019 & 1.9975 & 0.6963 \end{pmatrix}$$

Step 3. The singular value decomposition of $(L_o^D)^T L_c^D$:

$$[U, \Sigma, V] = SVD(L_o^T L_c)$$

gives

$$\Sigma = \text{diag} \begin{pmatrix} 5.3574 & 1.4007 & 0.1238 \end{pmatrix},$$

$$V = \begin{pmatrix} 0.8598 & -0.5055 & 0.0725 \\ 0.4368 & 0.6545 & -0.6171 \\ 0.2645 & 0.5623 & 0.7835 \end{pmatrix}.$$

Step 4.

$$\Sigma^{\frac{1}{2}} = \text{diag}(2.3146, 1.1835, 0.3519)$$

Step 5. The transforming matrix T is:

$$T = L_c V \Sigma^{-\frac{1}{2}}$$

$$= \begin{pmatrix} 0.9137 & -1.0506 & 0.5068 \\ 0.6257 & -0.1854 & -0.9419 \\ 0.3240 & 0.5475 & 0.4759 \end{pmatrix}$$

Step 6. The balanced matrices are:

$$\tilde{A} = T^{-1} A T = \begin{pmatrix} 0.5549 & 0.4098 & 0.0257 \\ -0.4098 & -0.1140 & 0.2629 \\ 0.0257 & -0.2629 & -0.4199 \end{pmatrix}$$

$$\tilde{B} = T^{-1} B = \begin{pmatrix} 1.8634 \\ 0.6885 \\ 0.0408 \end{pmatrix}$$

$$\tilde{C} = C T = (1.8634, -0.6885, 0.0408)$$

Verify.

$$T^{-1} C_G^D T^{-T} = T^T O_G^D T = \Sigma = \text{diag}(5.3574, 1.4007, 0.1238)$$

14.4 Model Reduction

Given an n -th order realization (A, B, C) with the transfer function matrix $G(\lambda) = C(\lambda I - A)^{-1}B$, where “ λ ” is complex variable “ s ” for the continuous-time case and is the complex variable $z = \frac{1+s}{1-s}$ for the discrete-time case, the **model reduction problem** is the problem of finding a reduced-order model (A_R, B_R, C_R) such that the associated transfer function matrix $G_R(\lambda)$ is as close as possible to $G(\lambda)$. We shall define the closeness here as having the error

$$\| G(\lambda) - G_R(\lambda) \|_{\infty}$$

as small as possible.

There are several approaches for solving the model reduction problem: the **Balanced Truncation Method**, the **Schur Method**, the **Hankel Norm Approximation Method**, the **Singular Perturbation Approximation Method**, and others.

We shall discuss the first three here. **Our discussion here is for the continuous-time case only;** the discrete-time case is analogous. For a discussion of the **singular perturbation method**, see [Exercise 23].

14.4.1 Model Reduction via Balanced Truncation

As the title suggests, the idea behind model reduction via balanced truncation is to truncate a balanced realization so that the truncated reduced-order model enjoys some of the important properties as of the original model. The idea is due to Moore (1981). The following Theorem forms a basis for that. Part (a) of the theorem is due to Pernebo and Silverman (1982) and Part (b) was proved by Enns (1984) and Glover (1984).

Theorem 14.4.1 (Stability and Error Bound of the Truncated Subsystem). *Let*

$$G(s) = \left[\begin{array}{cc|c} A_R & A_{12} & B_R \\ A_{21} & A_{22} & B_2 \\ \hline C_R & C_2 & 0 \end{array} \right] \quad (14.4.1)$$

be the transfer function matrix of an n -th order internally balanced stable system with the Grammian $\Sigma = \text{diag}(\Sigma_R, \Sigma_2)$, where

$$\begin{aligned} \Sigma_R &= \text{diag}(\sigma_1 I_{s_1}, \dots, \sigma_d I_{s_d}), \quad d < N \\ \Sigma_2 &= \text{diag}(\sigma_{d+1} I_{s_{d+1}}, \dots, \sigma_N I_{s_N}) \end{aligned} \quad (14.4.2)$$

and

$$\sigma_1 > \sigma_2 > \dots > \sigma_d > \sigma_{d+1} > \sigma_{d+2} > \dots > \sigma_N.$$

The multiplicity of σ_i is $s_i, i = 1, 2, \dots, N$ and $s_1 + s_2 + \dots + s_N = n$.

(a) Then the truncated system (A_R, B_R, C_R) with the transfer function

$$G_R(s) = \left[\begin{array}{c|c} A_R & B_R \\ \hline C_R & 0 \end{array} \right] \quad (14.4.3)$$

is balanced and stable.

(b) Furthermore, the error

$$\| G(s) - G_R(s) \|_\infty \leq 2(\sigma_{d+1} + \dots + \sigma_N). \quad (14.4.4)$$

In particular, if $d = N - 1$, then $\| G(s) - G_{N-1}(s) \|_\infty = 2\sigma_N$.

Proof: We leave the proof part (a) as an [Exercise 1]. We assume that the **singular values σ_i are distinct** and prove part (b) only in the case $n = N$. The proof in the general case can be easily done using the proof of this special case and is also left as an [Exercise 1]. The proof has been taken from Zhou, Doyle and Glover (1996, 158-160).

Let

$$\phi(s) = (sI - A_R)^{-1} \quad (14.4.5)$$

$$\psi(s) = sI - A_{22} - A_{21}\phi(s)A_{12} \quad (14.4.6)$$

$$\bar{B}(s) = A_{21}\phi(s)B_R + B_2 \quad (14.4.7)$$

$$\bar{C}(s) = C_R\phi(s)A_{12} + C_2 \quad (14.4.8)$$

Then

$$\begin{aligned} G(s) - G_R(s) &= C(sI - A)^{-1}B - C_R\phi(s)B_R \\ &= (C_R, C_2) \begin{pmatrix} sI - A_R & -A_{12} \\ -A_{21} & sI - A_{22} \end{pmatrix}^{-1} \begin{pmatrix} B_R \\ B_2 \end{pmatrix} - C_R\phi(s)B_R \\ &= \bar{C}(s)\psi^{-1}(s)\bar{B}(s) \end{aligned} \quad (14.4.9)$$

For $s = j\omega$, we have

$$\sigma_{max}[G(j\omega) - G_R(j\omega)] = \lambda_{max}^{\frac{1}{2}}[\psi^{-1}(j\omega)\bar{B}(j\omega)\bar{B}^*(j\omega)\psi^{-*}(j\omega)\bar{C}^*(j\omega)\bar{C}(j\omega)], \quad (14.4.10)$$

where $\lambda_{max}(M)$ denotes the largest eigenvalue of the matrix M .

Since the singular values are distinct, we have $\Sigma_2 = diag(\sigma_{r+1}, \dots, \sigma_n)$, and since Σ_2 satisfies

$$A_{22}\Sigma_2 + \Sigma_2 A_{22}^T + B_2 B_2^T = 0,$$

we obtain

$$\bar{B}(j\omega)\bar{B}^*(j\omega) = \psi(j\omega)\Sigma_2 + \Sigma_2\psi^*(j\omega).$$

Similarly, since Σ_2 also satisfies

$$\Sigma_2 A_{22} + A_{22}^T \Sigma_2 + C_2^T C_2 = 0,$$

we obtain

$$\bar{C}^*(j\omega)\bar{C}(j\omega) = \Sigma_2\psi(j\omega) + \psi^*(j\omega)\Sigma_2$$

Substituting these expressions of $\bar{B}(j\omega)\bar{B}^*(j\omega)$ and $\bar{C}^*(j\omega)\bar{C}(j\omega)$ into (14.4.10), we obtain after some algebraic manipulations

$$\sigma_{max}[G(j\omega) - G_R(j\omega)] = \lambda_{max}^{\frac{1}{2}}\{[\Sigma_2 + \psi^{-1}(j\omega)\Sigma_2\psi^*(j\omega)][\Sigma_2 + \psi^{-*}(j\omega)\Sigma_2\psi(j\omega)]\}$$

If $d = n - 1$, then $\Sigma_2 = \sigma_n$, and we immediately have

$$\sigma_{\max}[G(j\omega) - G_R(j\omega)] = \sigma_n \lambda_{\max}^{\frac{1}{2}} \{[1 + \Theta^{-1}(j\omega)][1 + \Theta(j\omega)]\}$$

where $\Theta = \psi^{-*}(j\omega)\psi(j\omega)$.

Note that $\Theta^{-*} = \Theta$ is a scalar function. So, $|\Theta(j\omega)| = 1$.

Using the triangle inequality, we then have

$$\sigma_{\max}[G(j\omega) - G_R(j\omega)] \leq \sigma_n [1 + |\Theta(j\omega)|] = 2\sigma_n$$

Thus we have proved the result for one-step; that is, we have proved the result assuming that the order of the truncated model is just one less than the original model. Using this “one-step” result, Theorem 14.4.1 can be proved for any order of the truncated model [**Exercise 1**]. ■

The above theorem shows that once a system is internally balanced, the balanced system can be truncated by eliminating the states corresponding to the less controllable and observable modes to obtain a reduced-order model that still preserves certain desirable properties of the original model (see Zhou et al. (1996)). In the following, we shall discuss how this can be done for the continuous-time system.

Choosing the order of the Reduced Model

Theorem 14.4.1 tells us that if d is chosen such that $\sigma_d >> \sigma_{d+1}$
or

$$\left(\sum_{i=1}^d \sigma_i^2 \right)^{\frac{1}{2}} >> \left(\sum_{i=d+1}^n \sigma_i^2 \right)^{\frac{1}{2}},$$

then the corresponding truncated model is expected to be a “good” one. In this case, the order q of the reduced-order model is

$$q = \sum_{i=1}^d s_i, \text{ where } s_i \text{ is the multiplicity of } \sigma_i.$$

Based on Theorem 14.4.1 and the above observation, we outline the following procedure for model reduction via balanced truncation.

Algorithm 14.4.1 (*Model Reduction via Balanced Truncation*)

Inputs: The system matrices A, B and C .

Outputs: The reduced-order model with the matrices A_R, B_R and C_R .

Assumption: A is stable.

Step 1. Find a balanced realization.

Step 2. Choose q , the order of model reduction, according to the criterion above.

Step 3. Partition the balanced realization in the form $A = \begin{pmatrix} A_R & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_R \\ B_2 \end{pmatrix}, C = (C_R, C_2)$, where A_R is of order q , and B_R and C_R are similarly defined.

■

The MATLAB function **modred** in the form

$$[\tilde{A}, \tilde{B}, \tilde{C}] = \text{modred}(A, B, C, \text{ELIM})$$

reduces the order of a matrix by eliminating the states specified in the vector ELIM.

Example 14.4.1 Consider Example 14.2.1 once more. Choose $q = 2$. Then A_R = The 2×2 leading principal submatrix of \tilde{A} is $\begin{pmatrix} -0.7659 & 0.5801 \\ -0.5801 & -2.4919 \end{pmatrix}$.

The eigenvalues of A_R are: -0.9900 , and -2.2678 .

Therefore A_R is stable.

The matrices B_R and C_R are: $B_R = \begin{pmatrix} 1.8602 \\ 0.6759 \end{pmatrix}$, $C_R = (1.8602, -0.6759)$.

Let $G_R(s) = C_R(sI - A_R)^{-1}B_R$.

Verification of the Error Bound: $\|G(s) - G_R(s)\|_\infty = 0.0012$. Since $2\sigma_3 = 0.0012$, the error bound given by (14.4.4) is satisfied.

14.4.2 The Schur Method for Model Reduction

The numerical difficulties of model reduction via balanced truncation using Algorithm 14.2.1 or Algorithm 14.2.2 (because of possible ill-conditioning of the transforming matrices) can be overcome if orthogonal matrices are used to transform the system to another **equivalent system** from which the reduced-order model is extracted. Safonov and Chiang (1989) have proposed a Schur method for this purpose.

A key observation here is that in Algorithm 14.2.1, the rows $\{1, \dots, d\}$ and rows $\{d+1, \dots, n\}$ of T^{-1} form bases for the left eigenspaces of the matrix $C_G O_G$ associated with the eigenvalues $\{\sigma_1^2, \dots, \sigma_d^2\}$ and $\{\sigma_{d+1}^2, \dots, \sigma_n^2\}$, respectively [Exercise 7].

Thus the idea will be to replace the matrices T and T^{-1} (which can be very ill-conditioned) by the orthogonal matrices (which are perfectly conditioned) sharing the same properties.

The Schur method, described below, constructs such matrices, using the real Schur form (RSF) of the matrix $C_G O_G$.

Specifically, the orthonormal bases for the right and left invariant subspaces corresponding to the “large” eigenvalues of the matrix $C_G O_G$ will be computed by finding the **ordered** real Schur form of $C_G O_G$.

Once the “large” eigenvalues are isolated from the “small” ones, the reduced order model preserving the desired properties, can be easily extracted.

We will now state the algorithm.

Algorithm 14.4.2 The Schur Method for Model Reduction (Continuous-time System).

Inputs:

A – The $n \times n$ state matrix

B – The $n \times m$ control matrix

C – The $r \times n$ output matrix

q – The dimension of the desired reduced-order model

Outputs:

\hat{A}_R – The $q \times q$ reduced state matrix

\hat{B}_R – The $q \times m$ reduced control matrix

\hat{C}_R – The $r \times q$ reduced output matrix.

S_1 and S_2 – Orthogonal transforming matrices such that $\hat{A}_R = S_1^T A S_2$,

$\hat{B}_R = S_1^T B$, and $\hat{C}_R = C S_2$

Assumption: A is stable.

Step 1. Compute the controllability Grammian C_G and the observability Grammian O_G by solving, respectively, the Lyapunov equations

$$AC_G + C_G A^T = -BB^T$$

and

$$A^T O_G + O_G A = -C^T C.$$

Step 2. Transform the matrix $C_G O_G$ to the real Schur form (RSF) Y ; that is, find an orthogonal matrix X such that $X^T C_G O_G X = Y$.

Note: The matrix $C_G O_G$ does not have any complex eigenvalues. Thus Y is actually an upper triangular matrix. Furthermore, the real eigenvalues are nonnegative.

Step 3. Reorder the eigenvalues of Y in ascending and descending order; that is, find orthogonal matrices U and V such that

$$U^T Y U = U^T X^T C_G O_G X U = U_S^T C_G O_G U_S = \begin{pmatrix} \lambda_1 & & * \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} \quad (14.4.11)$$

$$V^T Y V = V^T X^T C_G O_G X V = V_S^T C_G O_G V_S = \begin{pmatrix} \lambda_n & & * \\ & \ddots & \\ 0 & & \lambda_1 \end{pmatrix} \quad (14.4.12)$$

where $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.

(Note that $\lambda_n = \sigma_1^2, \lambda_{n-1} = \sigma_2^2, \dots$, and so on; where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$ are the Hankel singular values.)

Step 4. Partition the matrices U_S, V_S as follows:

$$U_S = (U_{1S}, U_{2S}), \quad V_S = (V_{1S}, V_{2S}).$$

Here U_{1S} contains the first $n - q$ columns of U_S and U_{2S} contains the remaining q columns. On the other hand, V_{1S} contains the first q columns of V_S and V_{2S} contains the remaining $n - q$ columns.

Note: Note that the columns of the V_{1S} and those of the matrix U_{1S} form, respectively, orthonormal bases for the right invariant subspace of $C_G O_G$ associated with the large eigenvalues $\{\sigma_1^2, \dots, \sigma_q^2\}$ and the small eigenvalues $\{\sigma_{q+1}^2, \dots, \sigma_n^2\}$. The columns of U_{2S} and V_{2S} , similarly, form orthonormal bases for the left invariant subspace of $C_G O_G$, with the large and the small eigenvalues, respectively.

Step 5. Find the SVD (the singular value decomposition) of $U_{2S}^T V_{1S}$:

$$Q\Sigma R^T = U_{2S}^T V_{1S}. \quad (14.4.13)$$

Step 6. Compute the transforming matrices:

$$S_1 = U_{2S} Q \Sigma^{-\frac{1}{2}} \quad (14.4.14)$$

$$S_2 = V_{1S} R \Sigma^{-\frac{1}{2}} \quad (14.4.15)$$

Step 7. Form the reduced-order matrices:

$$\begin{aligned} \hat{A}_R &= S_1^T A S_2, \\ \hat{B}_R &= S_1^T B, \\ \hat{C}_R &= C S_2. \end{aligned} \quad (14.4.16)$$

Flop-count: Since the reduction to the Schur Form using the QR iteration is an iterative process, an exact count cannot be given. The method just outlined requires approximately $100n^3$ flops.

Properties of the Reduced-order Model by the Schur-Method

The Schur method for model reduction does not give balanced realization. But the essential properties of the original model are preserved in the reduced-order model, as shown in the following Theorem.

Theorem 14.4.2 *The transfer function matrix $\hat{G}_R(s) = \hat{C}_R(sI - \hat{A}_R)^{-1}\hat{B}_R$ obtained by the Schur-method (Algorithm 14.4.2) is exactly the same as that of the one obtained via internal balancing (Algorithm 14.2.1). Furthermore, the controllability and observability Grammians of the reduced-order model are, respectively, given by:*

$$\hat{C}_G^R = S_1^T C_G S_1, \quad \hat{O}_G^R = S_2 O_G S_2.$$

Proof: We prove the first part and leave the second part as an [Exercise 9].

Let the transforming matrix T of the internal balancing algorithm and its inverse be partitioned as:

$$T = (T_1, T_2), \quad (14.4.17)$$

and

$$T^{-1} = \begin{pmatrix} T_I \\ * \end{pmatrix}, \quad (14.4.18)$$

where T_1 is $n \times d$ and T_I is of order $d \times n$.

Then the transfer function $G_R(s)$ of the reduced-order model obtained by Algorithm 14.2.1 is given by

$$G_R(s) = C_R(sI - A_R)^{-1}B_R = CT_1(sI - T_I A T_1)^{-1}T_I B \quad (14.4.19)$$

Again, the transfer function $\hat{G}_R(s)$ of the reduced-order model obtained by the Schur algorithm (Algorithm 14.4.2) is given by:

$$\hat{G}_R(s) = \hat{C}_R(sI - \hat{A}_R)^{-1}\hat{B}_R = CS_2(sI - S_1^T AS_2)^{-1}S_1^T B \quad (14.4.20)$$

The proof now amounts to establishing a relationship between S_1, S_2 and T_1 and T_I .

Let's define

$$V_R = (V_{1S}, U_{1S}), \quad (14.4.21)$$

$$\text{and } V_L = \begin{pmatrix} U_{2S}^T \\ V_{2S}^T \end{pmatrix}. \quad (14.4.22)$$

Then, since the first d and the last $(n-d)$ columns of T^{-1} , V_R , and V_L^{-1} span, respectively, the right eigenspaces associated with $\sigma_1^2, \dots, \sigma_d^2$ and $\sigma_{d+1}^2, \dots, \sigma_n^2$, it follows that there exist nonsingular matrices X_1 , and X_2 , such that

$$V_R = T \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} = V_L^{-1} \begin{pmatrix} E_1 & 0 \\ 0 & E_2 \end{pmatrix}. \quad (14.4.23)$$

From (14.4.21) and (14.4.23) we have

$$V_{1S} = T_1 X_1 \quad (14.4.24)$$

Thus

$$S_2 = V_{1S} R \Sigma^{-\frac{1}{2}} = T_1 X_1 R \Sigma^{-\frac{1}{2}} \quad (\text{using (14.4.24)}).$$

Similarly,

$$\begin{aligned}
S_1^T &= \Sigma^{-\frac{1}{2}} Q^T U_{2S}^T = \Sigma^{\frac{1}{2}} R^T (R\Sigma^{-1}Q^T) U_{2S}^T \\
&= \Sigma^{\frac{1}{2}} R^T (U_{2S}^T V_{1S})^{-1} U_{2S}^T \text{ (using (14.4.13))} \\
&= \Sigma^{\frac{1}{2}} R^T (U_{2S}^T T_1 X_1)^{-1} U_{2S}^T \text{ (using (14.4.24))} \\
&= \Sigma^{1/2} R^T X_1^{-1} T_I
\end{aligned}$$

Thus,

$$\begin{aligned}
\hat{G}_R(s) &= CS_2(sI - S_1^T AS_2)^{-1} S_1^T B \\
&= CT_1 X_1 R \Sigma^{-\frac{1}{2}} \left(sI - \Sigma^{\frac{1}{2}} R^T X_1^{-1} T_I A T_1 X_1 R \Sigma^{-\frac{1}{2}} \right)^{-1} \left(\Sigma^{\frac{1}{2}} R^T X_1^{-1} T_I B \right) \\
&= CT_1 (sI - T_I A T_1)^{-1} T_I B = G_R(s).
\end{aligned}$$

Note: Since $G_R(s)$ of Theorem 14.4.1 and $\hat{G}_R(s)$ of Theorem 14.4.2 are the same, from Theorem 14.4.1, we conclude that \hat{A}_R is stable and $\|G(s) - \hat{G}_R(s)\|_\infty \leq 2 \sum_{i=d+1}^n \sigma_i$, where σ_{d+1} through σ_n are the $(d+1)$ th through n th entries of the diagonal matrix Σ of the balancing algorithm; that is, σ 's are the Hankel singular values.

Relation to the Square-Root Method. There could be large round-off errors in explicit computation of the matrix product $C_G O_G$. The formation of the explicit product, however, can be avoided by computing the Cholesky factors L_c and L_o of the matrices C_G and O_G , using **Algorithm 8.6.1** described in Chapter 8. This then leads to the **square-root** method. We leave the derivation of the modified algorithm to the readers [**Exercise 10**]. For details, see Safonov and Chiang (1989).

Example 14.4.2

$$A = \begin{pmatrix} -1 & 2 & 3 \\ 0 & -2 & 1 \\ 0 & 0 & -3 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$$C = (1, 1, 1), q = 2.$$

The system (A, B, C) is **stable, controllable, and observable**.

Step 1. Solving the Lyapunov equations

$$AC_G + C_G A^T = -BB^T$$

and

$$A^T O_G + O_G A = -C^T C$$

we obtain

$$C_G = \begin{pmatrix} 3.9250 & 0.9750 & 0.4917 \\ 0.9750 & 0.3667 & 0.2333 \\ 0.4917 & 0.2333 & 0.1667 \end{pmatrix}$$

and

$$O_G = \begin{pmatrix} 0.5000 & 0.6667 & 0.7917 \\ 0.6667 & 0.9167 & 1.1000 \\ 0.7917 & 1.1000 & 1.3250 \end{pmatrix}.$$

Step 2. The Schur decomposition Y and the transforming matrix X obtained using the MATLAB function **schur**:

$$[X, Y] = \text{schur}(C_G O_G)$$

are

$$X = \begin{pmatrix} -0.9426 & -0.3249 & 0.0768 \\ -0.2885 & 0.6768 & -0.6773 \\ -0.1680 & 0.6606 & 0.7317 \end{pmatrix},$$

$$Y = \begin{pmatrix} 5.1028 & -5.2629 & -1.0848 \\ 0 & 0.0084 & 0.0027 \\ 0 & 0 & 0 \end{pmatrix}.$$

Step 3. Since the eigenvalues of Y are in decreasing order of magnitude, we take

$$V_S = X.$$

Next, we compute U_S such that the eigenvalues of $U_S^T C_G O_G U_S$ appear in increasing order of magnitude:

$$U_S = \begin{pmatrix} 0.2831 & -0.8579 & 0.4289 \\ -0.8142 & 0.0214 & 0.5802 \\ 0.5069 & 0.5134 & 0.6924 \end{pmatrix},$$

$$U_S^T C_G O_G U_S = \begin{pmatrix} 0 & -0.0026 & 0.8660 \\ 0 & 0.0084 & -5.3032 \\ 0 & 0 & 5.1028 \end{pmatrix}.$$

Step 4. Partitioning U_S and $V_S = X$, we obtain

$$U_{1S} = \begin{pmatrix} 0.2831 \\ -0.8142 \\ 0.5069 \end{pmatrix}$$

$$U_{2S} = \begin{pmatrix} -0.8579 & 0.4289 \\ 0.0214 & 0.5802 \\ 0.5134 & 0.6924 \end{pmatrix},$$

$$V_{1S} = \begin{pmatrix} -0.9426 & -0.3249 \\ -0.2885 & 0.6768 \\ -0.1680 & 0.6606 \end{pmatrix}, \quad V_{2S} = \begin{pmatrix} 0.0768 \\ -0.6773 \\ 0.7317 \end{pmatrix}$$

Step 5. The Singular value decomposition of the product $U_{2S}^T V_{1S}$ is given by:

$$[Q, \Sigma, R] = svd(U_{2S}^T V_{1S})$$

$$Q = \begin{pmatrix} 0.4448 & -0.8956 \\ -0.8956 & -0.4448 \end{pmatrix}, R = \begin{pmatrix} 0.9348 & -0.3553 \\ -0.3553 & -0.9348 \end{pmatrix} \text{ and } \Sigma = diag(1, 0.9441).$$

Step 6. The transforming matrices are:

$$S_1 = \begin{pmatrix} -0.7659 & 0.5942 \\ -0.5100 & -0.2855 \\ -0.3915 & -0.7903 \end{pmatrix} \text{ and } S_2 = \begin{pmatrix} -0.7659 & 0.6570 \\ -0.5099 & -0.5458 \\ -0.3916 & -0.5742 \end{pmatrix}.$$

Step 7. The matrices of the reduced order model are:

$$\hat{A}_R = S_1^T A S_2 = \begin{pmatrix} 0.3139 & 1.7204 \\ -1.9567 & -3.5717 \end{pmatrix}, \quad \hat{B}_R = S_1^T B = \begin{pmatrix} -1.6674 \\ -0.4817 \end{pmatrix},$$

and $\hat{C}_R = C S_2 = (-1.6676, -0.4631)$.

Verification of the Properties of the Reduced-order Model: We next verify that the reduced-order model has desirable properties such as stability and the error bound (14.4.4) is satisfied.

1. The eigenvalues of \hat{A}_R are: $\{-0.9900, -2.2678\}$. Thus \hat{A}_R is **stable**.

(Note that these eigenvalues are the same as those of A_R of order 2 obtained by Algorithm 14.4.1).

2. The controllability Grammian \hat{C}_G^R of the reduced order model is given by:

$$\hat{C}_G^R = S_1^T C_G S_1 = \begin{pmatrix} 3.5732 & -1.4601 \\ -1.4601 & 0.8324 \end{pmatrix}.$$

The eigenvalues of \hat{C}_G^R are 4.2053, and 0.2003. Thus \hat{C}_G^R is positive definite.

It is easily verified by solving the Lyapunov equation $\hat{A}_R \hat{C}_G^R + \hat{C}_G^R \hat{A}_R^T = -\hat{B}_R \hat{B}_R^T$,

that the \hat{C}_G^R given above is indeed the controllability Grammian of the reduced order model.

Similar results hold for the observability Grammian \hat{O}_{GR} .

Verification of the Error Bound

$$\|G(s) - \hat{G}_R(s)\|_\infty = 0.0012.$$

Since $2\sigma_3 = 0.0012$, the error bound (14.4.4) is verified.

Comparison of the Reduced order Models obtained by Balanced Truncation and the Schur Method with the original Model.

Figure 14.1 compares the errors of the reduced-order models with the theoretical error bound given by (14.4.4).

Figure 14.2 compares the step response of the orginal model with the step responses of the reduced-order models obtained by balanced truncation and the Schur method.

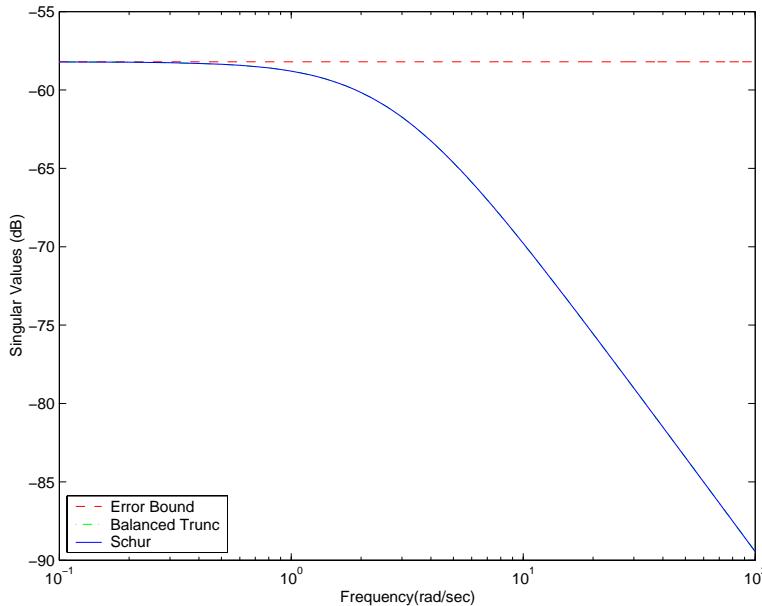


Figure 14.1: Theoretical Error Bound and Errors of the Reduced-order Models

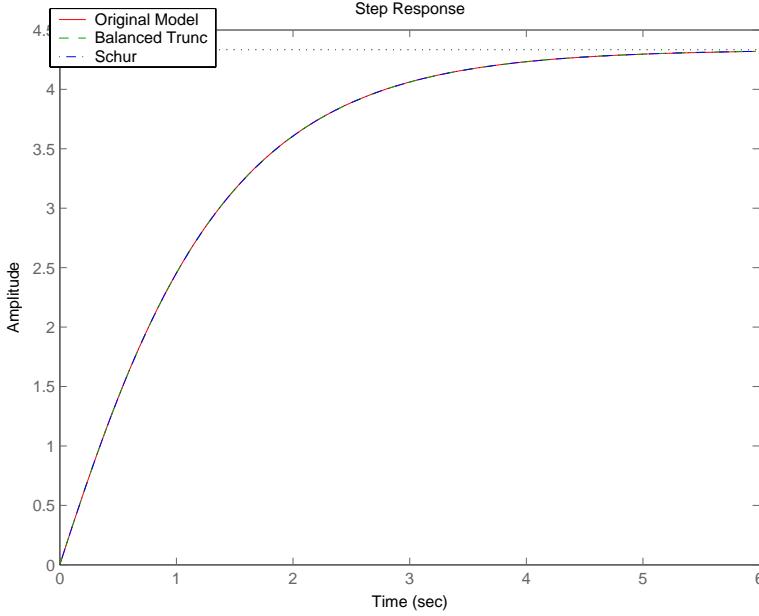


Figure 14.2: Step Responses of the Original and the Reduced-order Models

MATCONTROL NOTE: The MATCONTROL function **modreds** implements the Schur Algorithm (**Algorithm 14.4.2**) in the following format:

$$[A_R, B_R, C_R, S, T] = \text{modreds}(A, B, C, d).$$

The matrices A_R, B_R, C_R are the matrices of the reduced-order model of dimension d . The matrices S and T are the transforming matrices.

14.4.3 A Balancing-Free Square-Root Method for Model Reduction

By computing the matrices L and T a little differently than in the square-root method (**Algorithm 14.2.2**), the main advantages of the Schur method and the square-root method can be combined. The idea is due to Varga (1991).

Consider the **economy QR** factorizations of the matrices $L_c V_1$ and $L_o^T U_1$:

$$L_c V_1 = XW, \quad L_o^T U_1 = YZ,$$

where W and Z are nonsingular upper triangular and X and Y are orthonormal matrices.

Then L and T defined by

$$L = (Y^T X)^{-1} Y^T, \quad Z = X$$

are such that the system $(\tilde{A}, \tilde{B}, \tilde{C})$ with $\tilde{A} = LAZ$, $\tilde{B} = LB$ and $\tilde{C} = CZ$ form a **minimal realization** and therefore can be used to obtain a reduced-order model.

Example 14.4.3 Let's consider Example 14.2.2 once more.

Then $X = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}$, $Y = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}$, $W = -0.7071$, $Z = -0.7071$. The matrices L and Z in this case are

$$L = (-1 \ 0 \ 0), \ Z_2 \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}.$$

The matrices \tilde{A} , \tilde{B} and \tilde{C} are: $\tilde{A} = -1$, $\tilde{B} = -1$, $\tilde{C} = -1$.

14.5 Hankel-Norm Approximations

Let (A, B, C) be a stable realization of $G(s) = C(sI - A)^{-1}B$. Then the Hankel-norm of $G(s)$ is defined as

$$\| G(s) \|_H = \lambda_{\max}^{\frac{1}{2}}(C_G O_G), \quad (14.5.1)$$

where C_G and O_G are the controllability and observability Grammians, respectively, and $\lambda_{\max}(M)$ stands for the largest eigenvalue of M .

The **optimal Hankel-norm approximation problem** is the problem of finding an approximation $\hat{G}(s)$ of McMillan degree $k < n$ such that the norm of the error $\| G(s) - \hat{G}(s) \|_H$ is minimized.

The following theorem gives an achievable lower bound of the error of an approximation in Hankel-norm. Proof can be found in Glover (1984) or in Zhou, Doyle and Glover (1996, 189-190).

Theorem 14.5.1 Let $G(s)$ be a stable transfer function with Hankel singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k \geq \sigma_{k+1} \dots \geq \sigma_n > 0$. Then for all stable $\hat{G}(s)$ of McMillan degree $\leq k$

$$\| G(s) - \hat{G}(s) \|_H \geq \sigma_{k+1}.$$

■

We now give a result on characterization of all solutions to optimal Hankel-norm approximations and then state an algorithm to find an optimal Hankel-norm approximation.

The presentation here is based on Glover (1984). For proofs and other details, the readers are referred to the paper of Glover or the book by Zhou, Doyle and Glover (1996).

14.5.1 A Characterization of All Solutions to Hankel-Norm Approximation

The following theorem gives necessary and sufficient conditions for $\hat{G}(s)$ to be an optimal Hankel-norm approximation to $G(s)$. (For proof, see Glover (1984), Lemma 8.1).

Theorem 14.5.2 Let $G(s) = C(sI - A)^{-1}B$ be a stable, rational, square transfer function with singular values

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_k > \sigma_{k+1} = \sigma_{k+2} = \cdots = \sigma_{k+p} > \sigma_{k+p+1} \geq \cdots \geq \sigma_n > 0.$$

Let $\hat{G}(s)$ be of McMillan degree $k \leq n-p$. Then $\hat{G}(s)$ is an optimal Hankel-norm approximation to $G(s)$ if and only if there exists $(\hat{A}, \hat{B}, \hat{C}), P_e, Q_e$ such that

(a) $\hat{G}(s)$ is the stable part of

$$\hat{C}(sI - \hat{A})^{-1}\hat{B} \quad (14.5.2)$$

(b) The matrices P_e and Q_e satisfy

$$(i) \quad A_e P_e + P_e A_e^T + B_e B_e^T = 0 \quad (14.5.3)$$

$$(ii) \quad A_e^T Q_e + Q_e A_e + C_e^T C_e = 0 \quad (14.5.4)$$

$$(iii) \quad P_e Q_e = \sigma_{k+1}^2 I \quad (14.5.5)$$

where A_e, B_e and C_e are defined by

$$A_e = \begin{pmatrix} A & 0 \\ 0 & \hat{A} \end{pmatrix}, \quad B_e = \begin{pmatrix} B \\ \hat{B} \end{pmatrix}, \quad C_e = (C, -\hat{C}). \quad (14.5.6)$$

(c) If P_e and Q_e are partitioned conformally with A_e in (14.5.6) as:

$$P_e = \begin{pmatrix} P_{11} & P_{12} \\ P_{12}^T & P_{22} \end{pmatrix}, \quad Q_e = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{pmatrix},$$

then

$$In(P_{22}) = In(Q_{22}) = (k, l, 0). \quad (14.5.7)$$

Further, $\dim(\hat{A}) = k + l$ can be chosen $\leq n + 2k - 1$.

■

We now give a construction of $\hat{A}, \hat{B}, \hat{C}$ that satisfy the equations (14.5.3)-(14.5.7) for a **balanced realization** (A, B, C) of $G(s)$, which will be the basis of an algorithm for Hankel-norm approximation. The construction, however, remains valid for a more general class of realization, and the details can be found in Glover (1984) (**Theorem 6.3**).

Theorem 14.5.3 Let (A, B, C) be a balanced realization of $G(s)$ with the matrix of the singular values

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n),$$

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > \sigma_{k+1} = \sigma_{k+2} = \dots = \sigma_{k+p} > \sigma_{k+p+1} \geq \dots \geq \sigma_n > 0$$

Partition $\Sigma = (\Sigma_1, \sigma_{k+1} I_p)$, $\sigma_{k+1} \neq 0$, and then partition A, B, C conformally:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}, \quad C = (C_1, C_2). \quad (14.5.8)$$

Define now

$$\hat{A} = \Gamma^{-1}(\sigma_{k+1}^2 A_{11}^T + \Sigma_1 A_{11} \Sigma_1 - \sigma_{k+1} C_1^T U B_1^T) \quad (14.5.9)$$

$$\hat{B} = \Gamma^{-1}(\Sigma_1 B_1 + \sigma_{k+1} C_1^T U) \quad (14.5.10)$$

$$\hat{C} = C_1 \Sigma_1 + \sigma_{k+1} U B_1^T \quad (14.5.11)$$

where

$$\Gamma = \Sigma_1^2 - \sigma_{k+1}^2 I \quad (14.5.12)$$

and U is such that

$$B_2 = -C_2^T U. \quad (14.5.13)$$

Then A_e, B_e and C_e defined by (14.5.6) satisfy (14.5.3)-(14.5.5), with

$$P_e = \begin{pmatrix} \Sigma_1 & 0 & I \\ 0 & \sigma_{k+1} I & 0 \\ I & 0 & \Sigma_1 \Gamma^{-1} \end{pmatrix} \quad (14.5.14)$$

$$Q_e = \begin{pmatrix} \Sigma_1 & 0 & -\Gamma \\ 0 & \sigma_{k+1} I & 0 \\ -\Gamma & 0 & \Sigma_1 \Gamma \end{pmatrix}. \quad (14.5.15)$$

■

Based on Theorem 14.5.2 and Theorem 14.5.3 the following algorithm can now be written down for finding a Hankel-norm approximation of a balanced realization (A, B, C) of $G(s)$.

Algorithm 14.5.1 An Algorithm for Hankel-norm Approximation of a Continuous-time System

Inputs:

1. The matrices A, B and C of a realization $G(s)$.
2. k -McMillan degree of Hankel-norm approximation

Outputs: The matrices \hat{A}_{11}, \hat{B}_1 and \hat{C}_1 of a Hankel-norm approximation $\hat{G}(s)$ of McMillan degree k .

Assumptions:

1. A is stable.
2. The Hankel singular values σ_i are such that $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > \sigma_{k+1} > \sigma_{k+2} \geq \dots \geq \sigma_n > 0$.

Step 1. Find a balanced realization $(\tilde{A}, \tilde{B}, \tilde{C})$ of $G(s)$ using Algorithm 14.2.1 or Algorithm 14.2.2, whichever is appropriate.

Step 2. Partition $\Sigma = \text{diag}(\Sigma_1, \sigma_{k+1})$ and then order the balanced realization $(\tilde{A}, \tilde{B}, \tilde{C})$ conformally so that

$$\tilde{A} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad \tilde{B} = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}, \quad \tilde{C} = (C_1, C_2).$$

(Note that A_{11} is $(n-1) \times (n-1)$).

Step 3. Compute the matrix U satisfying (14.5.13) and form the matrices Γ, \hat{A}, \hat{B} and \hat{C} using the equations (14.5.9)-(14.5.12).

Step 4. Block diagonalize \hat{A} to obtain \hat{A}_{11} :

- (a) Transform \hat{A} to an upper real Schur form and then order the real Schur form so that the eigenvalues with negative real parts appear first; that is, find an orthogonal matrix V_1 such that $V_1^T \hat{A} V_1$ is in upper real Schur form and then find another orthogonal matrix V_2 such that

$$V_2^T V_1^T \hat{A} V_1 V_2 = \begin{pmatrix} \hat{A}_{11} & \hat{A}_{12} \\ 0 & \hat{A}_{22} \end{pmatrix} \quad (14.5.16)$$

where the eigenvalues of \hat{A}_{11} have negative real parts and those of \hat{A}_{22} have positive real parts.

- (b) Solve the Sylvester equation for $X \in \mathbb{R}^{k \times (n-k-1)}$:

$$\hat{A}_{11} X - X \hat{A}_{22} + \hat{A}_{12} = 0 \quad (14.5.17)$$

- (c) Let

$$T = V_1 V_2 \begin{pmatrix} I & X \\ 0 & I \end{pmatrix} = (T_1, T_2) \quad (14.5.18)$$

$$S = \begin{pmatrix} I & -X \\ 0 & I \end{pmatrix} V_2^T V_1^T = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix}. \quad (14.5.19)$$

Step 5. Form

$$\hat{B}_1 = S_1 \hat{B} \quad (14.5.20)$$

$$\hat{C}_1 = \hat{C} T_1 \quad (14.5.21)$$

■

Example 14.5.1 Consider Example 14.2.1 once more. Then

Step 1. The balanced matrices \tilde{A} , \tilde{B} , and \tilde{C} are the same as of Example 14.2.1.

Step 2.

$$\Sigma = \text{diag}(2.2589, 0.0917, 0.0006).$$

$$\Sigma_1 = \text{diag}(2.2589, 0.0917).$$

$k = 2$, and $\sigma_3 = 0.0006$.

$$A_{11} = \begin{pmatrix} -0.7659 & 0.5801 \\ -0.5801 & -2.4919 \end{pmatrix},$$

$$B_1 = \begin{pmatrix} 1.8602 \\ 0.6759 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0.0581 \end{pmatrix}$$

$$C_1 = (1.8602, -0.6759), \quad C_2 = (-0.0581).$$

Step 3.

$$U = 1, \quad \Gamma = \text{diag}(5.1028, 0.0084)$$

$$\hat{A} = \begin{pmatrix} -0.7659 & 0.0235 \\ -14.2961 & -2.4919 \end{pmatrix}$$

$$\hat{B} = \begin{pmatrix} -0.8235 \\ -7.3735 \end{pmatrix}, \quad \hat{C} = (-4.2020, -0.0620).$$

Step 4.

(a) \hat{A} is already in upper Schur form. Thus $V_1 = I$.

The eigenvalues of \hat{A} are $-0.9900, -2.2678$. Since both have negative real parts, no reordering is needed.

Thus, $V_2 = I$, $\hat{A}_{11} = \hat{A}$, $\hat{A}_{12} = 0$, $\hat{A}_{22} = 0$.

(b) $X = 0$

(c)

$$T = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}, T_1 = I, T_2 = I.$$

$$S = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}, S_1 = I, S_2 = I.$$

Step 5.

$$\hat{B}_1 = \hat{B}, \hat{C}_1 = \hat{C}$$

Obtaining an Error BoundNext, we show how to construct a matrix \hat{D}_1 such that with

$$\hat{G}(s) = \hat{D}_1 + \hat{C}_1(sI - \hat{A}_{11})^{-1}\hat{B}_1$$

an error bound for the approximation $\|G(s) - \hat{G}(s)\|_\infty$ can be obtained.

Define

$$\hat{B}_2 = S_2\hat{B}, \hat{C}_2 = \hat{C}T_2, \hat{D}_1 = -\sigma_{k+1}U. \quad (14.5.22)$$

Step 6. Update now \hat{D}_1 as follows:**6.1.** Find a balanced realization of the system $(-\hat{A}_{22}, \hat{B}_2, \hat{C}_2, \hat{D}_1)$; say (A_3, B_3, C_3, D_3) .

Compute the Hankel singular values of this balanced system and call them

$$\mu_1, \mu_2, \dots, \mu_{n-k-1}.$$

6.2. Let q be an integer greater than or equal to $r + m$, where r and m are the number of outputs and inputs, respectively. Define $Z, Y \in \mathbb{R}^{q \times (n-k-1)}$ by

$$Z = \begin{pmatrix} B_3^T \\ 0 \end{pmatrix}, Y = \begin{pmatrix} C_3 \\ 0 \end{pmatrix}.$$

Denote the i th columns of Z and Y by z_i and y_i , respectively.**6.3.** For $i = 1, 2, \dots, n - k - 1$ do(i) Find Householder matrices H_1 and H_2 such that

$$H_1 y_i = - \begin{pmatrix} \alpha & 0 & \dots & 0 \end{pmatrix}^T$$

and

$$H_2 z_i = - \begin{pmatrix} \beta & 0 & \dots & 0 \end{pmatrix}^T.$$

(ii) Define

$$U_i = H_1 \begin{pmatrix} -\alpha/\beta & 0 & 0 & 0 \\ 0 & 0 & I_{r-1} & 0 \\ 0 & I_{m-1} & 0 & 0 \\ 0 & 0 & 0 & I_{q-r-m+1} \end{pmatrix} H_2$$

(iii) If $i < n - k + 1$, then For $j = i + 1$ to $(n - k + 1)$ do

$$y \equiv -(y_j \mu_j + U z_j \mu_i)(\mu_i^2 - \mu_j^2)^{-1/2}$$

$$z_j \equiv (z_j \mu_j + U^T y_j \mu_i)(\mu_i^2 - \mu_j^2)^{-1/2}$$

$$y_j = y$$

$$(iv) \text{ Compute } \hat{D}_1 = \hat{D}_1 + (-1)^i \mu_i \begin{pmatrix} I_r & 0 \end{pmatrix} U \begin{pmatrix} I_m \\ 0 \end{pmatrix}$$

Theorem 14.5.4 (An Error Bound):

$$\|G(s) - \hat{G}(s)\|_\infty \leq \sigma_{k+1} + \mu_1 + \mu_2 + \dots + \mu_{n-k-1}$$

Example 14.5.2 Consider $k = 2$ and

$$A = \begin{pmatrix} -1 & 2 & -1 & 3 \\ 0 & -2 & 2 & 0 \\ 0 & 0 & -3 & -2 \\ 0 & 0 & 0 & -4 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & -2 \\ 2 & 0 \\ -1 & 5 \\ 2 & 3 \end{pmatrix}, \quad C = \begin{pmatrix} -1 & 0 & 2 & -3 \\ 1 & 1 & -2 & 1 \end{pmatrix}$$

Step 1: The Hankel singular values:

$$\{ 4.7619 \quad 1.3650 \quad 0.3614 \quad 0.0575 \}$$

Step 2:

$$\tilde{A} = \begin{pmatrix} -1.1663 & 1.7891 & -0.2132 & 0.3266 \\ -0.0919 & -2.5711 & 0.7863 & -1.0326 \\ 0.3114 & 2.6349 & -3.7984 & 1.5960 \\ -0.3641 & 0.5281 & -0.2582 & -2.4642 \end{pmatrix}, \quad \tilde{B} = \begin{pmatrix} 3.3091 & -0.3963 \\ -0.2903 & 2.6334 \\ -0.6094 & -1.5409 \\ 0.4947 & -0.1967 \end{pmatrix}$$

$$\tilde{C} = \begin{pmatrix} -2.4630 & 1.3077 & 0.9011 & 0.1600 \\ 2.2452 & -2.3041 & 1.3905 & -0.5078 \end{pmatrix}$$

The permutation matrix that does the reordering is

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The reordered balanced realization gives

$$\Sigma_1 = \begin{pmatrix} 4.7619 & 0 & 0 \\ 0 & 1.3650 & 0 \\ 0 & 0 & 0.0575 \end{pmatrix}, \quad \Sigma_2 = 0.3614 = \sigma_3.$$

$$\begin{aligned}
A_{11} &= \begin{pmatrix} -1.1663 & 1.7891 & 0.3266 \\ -0.0919 & -2.5711 & -1.0326 \\ -0.3641 & 0.5281 & -2.4642 \end{pmatrix}, \quad A_{12} = \begin{pmatrix} -0.2132 \\ 0.7863 \\ -0.2582 \end{pmatrix} \\
A_{21} &= \begin{pmatrix} 0.3114 & 2.6349 & 1.5960 \end{pmatrix}, \quad A_{22} = -3.7984 \\
B_1 &= \begin{pmatrix} 3.3091 & -0.3963 \\ -0.2903 & 2.6334 \\ 0.4947 & -0.1967 \end{pmatrix}, \quad B_2 = \begin{pmatrix} -0.6094 & -1.5409 \end{pmatrix} \\
C_1 &= \begin{pmatrix} -2.4630 & 1.3077 & 0.1600 \\ 2.2452 & -2.3041 & -0.5078 \end{pmatrix}, \quad C_2 = \begin{pmatrix} 0.9011 \\ 1.3905 \end{pmatrix}
\end{aligned}$$

Step 3:

$$\begin{aligned}
U &= \begin{pmatrix} 0.2000 & 0.5057 \\ 0.3086 & 0.7804 \end{pmatrix} \\
\Gamma &= \begin{pmatrix} 22.5447 & 0. & 0. \\ 0. & 1.7326 & 0. \\ 0. & 0. & -0.1273 \end{pmatrix}, \quad \hat{A} = \begin{pmatrix} -1.1872 & 0.4948 & -0.0019 \\ 0.0063 & -2.3615 & 0.0072 \\ 0.3687 & 1.5211 & 2.5932 \end{pmatrix} \\
\hat{B} &= \begin{pmatrix} -0.7022 & 0.0756 \\ 0.3225 & -1.8376 \\ 0.1306 & 0.9841 \end{pmatrix}, \quad \hat{C} = \begin{pmatrix} 11.5617 & -2.2454 & 0.0090 \\ -10.9487 & 2.4347 & -0.0295 \end{pmatrix}
\end{aligned}$$

Step 4:

$$\begin{aligned}
V_1 &= \begin{pmatrix} -0.3754 & -0.9222 & -0.0930 \\ 0.8936 & -0.3335 & -0.3006 \\ -0.2462 & 0.1960 & -0.9492 \end{pmatrix}, \quad V_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
\hat{A}_{11} &= \begin{pmatrix} -2.3661 & 0.4343 \\ 0 & -1.1847 \end{pmatrix}, \quad \hat{A}_{12} = \begin{pmatrix} 1.3682 \\ -0.7792 \end{pmatrix}, \quad \hat{A}_{22} = 2.5953.
\end{aligned}$$

Solution of (14.5.17) gives

$$X = \begin{pmatrix} 0.2577 \\ -0.2061 \end{pmatrix}$$

and then

$$\begin{aligned}
T_1 &= \begin{pmatrix} -0.3754 & -0.9222 \\ 0.8936 & -0.3335 \\ -0.2462 & 0.1960 \end{pmatrix}, \quad T_2 = \begin{pmatrix} 0.0003 \\ -0.0015 \\ -1.0530 \end{pmatrix} \\
S_1 &= \begin{pmatrix} -0.3515 & 0.9710 & -0.0015 \\ -0.9413 & -0.3954 & 0.0003 \end{pmatrix}, \quad S_2 = \begin{pmatrix} -0.0930 & -0.3006 & -0.9492 \end{pmatrix}
\end{aligned}$$

Step 5: Using (14.5.20)-(14.5.22), we obtain

$$\hat{B}_1 = \begin{pmatrix} 0.5597 & -1.8124 \\ 0.5335 & 0.6558 \end{pmatrix}, \quad \hat{C}_1 = \begin{pmatrix} -6.3494 & -9.9113 \\ 6.2934 & 9.2788 \end{pmatrix}$$

$$\hat{B}_2 = \begin{pmatrix} -0.1556 & -0.3889 \end{pmatrix}, \hat{C}_2 = 10^{-1} \begin{pmatrix} -0.0237 \\ 0.2383 \end{pmatrix}, \hat{D}_1 = \begin{pmatrix} -0.0723 & -0.1828 \\ -0.1115 & -0.2820 \end{pmatrix}$$

Step 6.1: The matrices of the balanced realization of the system $(-\hat{A}_{22}, \hat{B}_2, \hat{C}_2, \hat{D}_1)$ are:

$$A_3 = \begin{pmatrix} -2.5953 \end{pmatrix}, B_3 = 10^{-1} \begin{pmatrix} -0.3720 & -0.9298 \end{pmatrix}$$

$$C_3 = 10^{-1} \begin{pmatrix} -0.0991 \\ 0.9966 \end{pmatrix}, D_3 = \begin{pmatrix} -0.0723 & -0.1828 \\ -0.1115 & -0.2820 \end{pmatrix}.$$

The system (A_3, B_3, C_3, D_3) has only one Hankel Singular value $\mu_1 = 0.0019$.

Step 6.2: Taking $q = r + m = 4$, we obtain

$$Z = 10^{-1} \begin{pmatrix} -0.3720 \\ -0.9298 \\ 0. \\ 0. \end{pmatrix}, Y = 10^{-1} \begin{pmatrix} -0.0991 \\ 0.9966 \\ 0. \\ 0. \end{pmatrix}$$

Step 6.3: $i = 1, \alpha = \beta = .1001$

$$H_1 = \begin{pmatrix} -0.0989 & 0.9951 & 0. & 0. \\ 0.9951 & 0.0989 & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}, H_2 = \begin{pmatrix} 0.3714 & 0.9285 & 0. & 0. \\ 0.9285 & -0.3714 & 0. & 0. \\ 0. & 0. & -1. & 0. \\ 0. & 0. & 0. & -1. \end{pmatrix}$$

$$U_1 = \begin{pmatrix} 0.0368 & 0.0919 & -0.9951 & 0. \\ -0.3696 & -0.9239 & -0.0989 & 0. \\ 0.9285 & -0.3714 & 0. & 0. \\ 0. & 0. & 0. & -1. \end{pmatrix}, \hat{D}_1 = \begin{pmatrix} -0.0723 & -0.1829 \\ -0.1108 & -0.2803 \end{pmatrix}$$

Verification: Let $\hat{G}(s) = \hat{C}_1(sI - \hat{A}_{11})^{-1}\hat{B}_1 + \hat{D}_1$. Then

$$0.3627 = \|G(s) - \hat{G}(s)\|_\infty \leq \sigma_3 + \mu_1 = 0.3633.$$

Also, $\|G(s) - \hat{G}(s)\|_H = 0.3614 = \sigma_3$.

Figure 14.3 below compares the step response of the original model with that of the Hankel-norm approximation model.

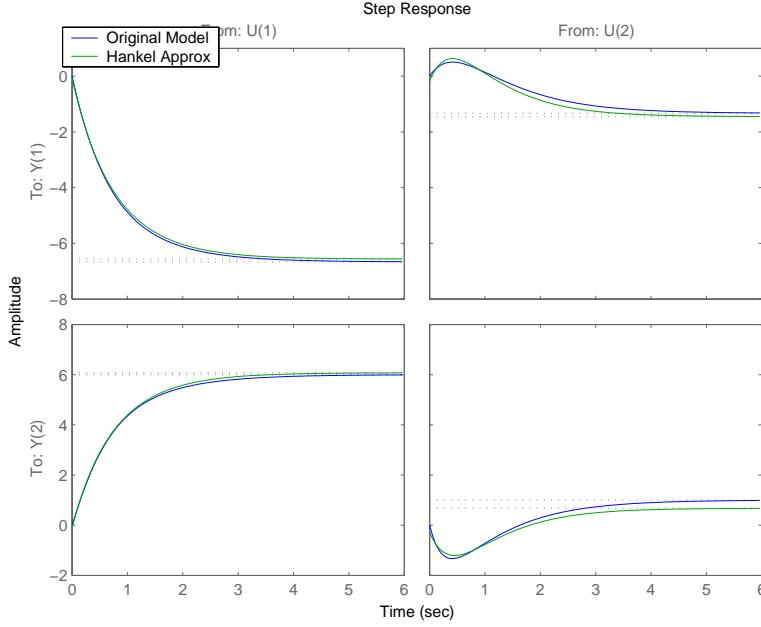


Figure 14.3: **Step Responses of the Original and Hankel Norm Approximation Models**

MATCONTROL NOTE: Algorithm 14.5.1 has been implemented in Matcontrol function **hnaprx**.

14.6 Model Reduction of an Unstable System

We have so far considered model reduction of a stable system.

However, model reduction of an unstable system can also be performed. Varga (2001) has proposed two approaches. The first approach consists of finding only the reduced-order model of the stable part and then including the unstable part in the resulting reduced model. The second approach is based on computing a stable rational coprime factorization of the transfer function matrix and then reducing the stable system. We describe just the first approach here. For details of the second approach, see Varga (2001).

Step 1. Decompose the transfer function matrix $G(\lambda)$ additively as:

$$G(\lambda) = G_S(\lambda) + G_U(\lambda)$$

such that $G_S(\lambda)$ is the stable part and $G_U(\lambda)$ is the unstable part.

Step 2. Find a reduced-order model $G_{RS}(\lambda)$ of the stable part $G_S(\lambda)$.

Step 3. The reduced-order model $G_R(\lambda)$ of $G(\lambda)$ is then given by

$$G_R(\lambda) = G_{RS}(\lambda) + G_U(\lambda).$$

Computational Remarks. The decomposition in Step 1 can be performed by block-diagonalizing the matrix A using the procedure of Step 4 of Algorithm 14.5.1.

14.7 Frequency Weighted Model Reduction

In this section, we consider the frequency weighted model reduction, proposed by Enns (1984). Specifically, the following problem is considered.

Given a stable transfer function matrix $G(s) = C(sI - A)^{-1}B$ and the two input and output weighting transfer function matrices $W_i = C_i(sI - A_i)^{-1}B_i$, and $W_o = C_o(sI - A_o)^{-1}B_o$, find a reduced-order model

$$G_R(s) = C_R(sI - A_R)^{-1}B_R$$

such that $\| W_o(G - G_R)W_i \|_\infty$ is as small as possible.

The effect of weighting on the model reduction is the possible reduction of the errors at the high frequencies.

The weighting model reduction problem can be solved in a similar way as the model reduction procedure by balanced truncation described in Section 14.4.

First, we note that the state space realization for the weighted transfer matrix is given by

$$W_o G W_i = \bar{C}(sI - \bar{A})^{-1}\bar{B},$$

where

$$\bar{A} = \begin{pmatrix} A & 0 & BC_i \\ B_o C & A_o & 0 \\ 0 & 0 & A_i \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} 0 \\ 0 \\ B_i \end{pmatrix}, \quad \bar{C} = (0, C_o, 0). \quad (14.7.1)$$

Let \hat{C}_G and \hat{O}_G be the solutions to the Lyapunov equations:

$$\bar{A}\hat{C}_G + \hat{C}_G(\bar{A})^T + \bar{B}\bar{B}^T = 0$$

$$\hat{O}_G\bar{A} + (\bar{A})^T\hat{O}_G + (\bar{C})^T\bar{C} = 0$$

Then the input weighted Grammian \bar{C}_G and the output weighted Grammian \bar{O}_G are defined by

$$\bar{C}_G \equiv (I_n, 0)\hat{C}_G \begin{pmatrix} I_n \\ 0 \end{pmatrix}$$

and

$$\bar{O}_G \equiv (I_n, 0)\hat{O}_G \begin{pmatrix} I_n \\ 0 \end{pmatrix}.$$

It can be shown [Exercise 21] that \bar{C}_G and \bar{O}_G satisfy:

$$\begin{aligned} \begin{pmatrix} A & BC_i \\ 0 & A_i \end{pmatrix} \begin{pmatrix} \bar{C}_G & \bar{C}_{G_{21}} \\ \bar{C}_{G_{21}}^T & \bar{C}_{G_{22}} \end{pmatrix} + \begin{pmatrix} \bar{C}_G & \bar{C}_{G_{21}} \\ \bar{C}_{G_{21}}^T & \bar{C}_{G_{22}} \end{pmatrix} \begin{pmatrix} A^T & 0 \\ C_i^T B^T & A_i^T \end{pmatrix} \\ + \begin{pmatrix} 0 \\ B_i \end{pmatrix} (0 \ B_i^T) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{aligned} \quad (14.7.2)$$

$$\begin{pmatrix} \bar{O}_G & \bar{O}_{G_{12}} \\ \bar{O}_{G_{12}}^T & \bar{O}_{G_{22}} \end{pmatrix} \begin{pmatrix} A & 0 \\ B_o C & A_o \end{pmatrix} + \begin{pmatrix} A^T & C^T B_o^T \\ 0 & A_o^T \end{pmatrix} \begin{pmatrix} \bar{O}_G & \bar{O}_{G_{12}} \\ \bar{O}_{G_{12}}^T & \bar{O}_{G_{22}} \end{pmatrix} + \begin{pmatrix} 0 \\ C_o^T \end{pmatrix} (0 \ C_o) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}. \quad (14.7.3)$$

Consider now two special cases.

Case 1. $W_i = I$. Then \bar{C}_G can be obtained from

$$\bar{C}_G A^T + A \bar{C}_G + B B^T = 0.$$

Case 2. $W_o = I$. Then \bar{O}_G can be obtained from

$$\bar{O}_G A + A^T \bar{O}_G + C^T C = 0.$$

Now, let T be a nonsingular matrix such that

$$T \bar{C}_G T^T = (T^{-1})^T \bar{O}_G T^{-1} = \text{diag}(\bar{\Sigma}_1, \bar{\Sigma}_2); \quad (14.7.4)$$

that is, the matrix T makes the realization balanced.

Let $\bar{\Sigma}_1 = \text{diag}(\sigma_1 I_{s_1}, \dots, \sigma_r I_{s_r})$ and $\bar{\Sigma}_2 = \text{diag}(\sigma_{r+1} I_{s_{r+1}}, \dots, \sigma_n I_{s_n})$.

Partition the system (TAT^{-1}, TB, CT^{-1}) accordingly; that is

$$TAT^{-1} = \begin{pmatrix} \bar{A}_R & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \end{pmatrix}, \quad TB = \begin{pmatrix} \bar{B}_R \\ \bar{B}_2 \end{pmatrix}$$

and

$$CT^{-1} = (\bar{C}_R, \bar{C}_2).$$

Then $(\bar{A}_R, \bar{B}_R, \bar{C}_R)$ is a weighted reduced-order model.

If the full-order original model is minimal, then $\bar{\Sigma}_1 > 0$.

Unfortunately, the stability of the reduced-order model here cannot, in general, be guaranteed.

However, there are some special cases of weightings for which the reduced-order models are stable [**Exercise 22**].

14.8 Summary and Comparisons of Model Reduction Procedures

We have described the following techniques for model reduction of a stable system:

- (i) The balanced truncation procedure (**Algorithm 14.4.1**)
- (ii) The Schur method (**Algorithm 14.4.2**)
- (iii) The Hankel-norm approximation algorithm (**Algorithm 14.5.1**)

(iv) Frequency weighted model reduction (**Section 14.7**)

For the first three methods (i)-(iii), the error satisfies

$$\| G(s) - G_R(s) \|_{\infty} \leq 2(\sigma_{d+1} + \sigma_{d+2} + \dots + \sigma_n),$$

where $\sigma_1, \sigma_2, \dots, \sigma_n$ are the Hankel singular values and d is such that the states corresponding to σ_{d+1} through σ_n are less controllable and observable than those corresponding to $\sigma_1, \dots, \sigma_d$. Furthermore, for an optimal Hankel-norm approximation, the reduced-order model $G_R(s)$ has the property: $\inf \| G - G_R(s) \|_H = \sigma_{k+1}$, where $G_R(s)$ is of McMillan degree k .

The weighted model reduction procedure in Section 14.7 does not enjoy any of the above properties. Even the stability in general cannot be guaranteed. Stability, however, in some special cases can be proved. See Enns (1984) for details.

If the system is not stable, model reduction is still possible using the three simple steps of Section 14.6.

The balanced truncation procedure for model reduction and Algorithm 14.5.1 need computation of a balanced realization. Two algorithms (**Algorithm 14.2.1** and **Algorithm 14.2.2**) have been described for this purpose. Both these algorithms suffer from the danger of possible ill-conditioning of the transforming matrices. **However, the methods usually work well in practice for well-equilibrated systems.**

The Schur method has been designed to avoid such possible ill-conditioning.

Unfortunately, because of the requirement of explicitly computing the product of the controllability and observability Grammians, *the Schur method is usually less accurate for moderately ill-conditioned systems than the square-root method* (see Varga (2001)). The main advantages of the balanced truncation procedure and the Schur method have been combined in the **balanced-free square-root method** by Varga (1991). Numerical experiments performed by Varga (2001) show that the accuracy of this method is usually better than either of the Schur methods or the balanced truncation method using the square-root algorithm for balancing.

Finally, we remark that it is **very important that the system be scaled properly for the application of the balanced-truncation or the Hankel-norm approximation method**. One way to do this is to attempt to reduce the 1-norm of the scaled system matrix

$$S = \begin{pmatrix} Z^{-1}AZ & Z^{-1}B \\ CZ & 0 \end{pmatrix},$$

where Z is a positive definite matrix.

Note that the Hankel singular values are not affected by such a coordinate transformation; in particular, by coordinate scaling of diagonal matrices.

For a comparative study of different model reduction algorithms and detailed description of available software, see Varga (2001).

14.9 Some Selected Software

14.9.1 MATLAB CONTROL SYSTEM TOOLBOX

State-space models

- balreal - Grammian-based input/output balancing
- modred - Model state reduction

14.9.2 MATCONTROL

- BALSVD - Internal balancing using the singular value decomposition
- BALSQT - Internal balancing using the square-root algorithm
- MODREDS - Model reduction using the Schur method
- HNAPRX - Hankel norm approximation

14.9.3 CSP-ANM

Model Reduction

- The Schur method for model reduction is implemented as `DominantSubsystem [system, Method → SchurDecomposition]`.
- The square root method for model reduction is implemented as `DominantSubsystem [system, Method → SquareRoot]`.

14.9.4 SLICOT

Model Reduction

- AB09AD Balance and Truncate model reduction
- AB09BD Singular perturbation approximation based model reduction
- AB09CD Hankel norm approximation based model reduction
- AB09DD Singular perturbation approximation formulas
- AB09ED Hankel norm approximation based model reduction of unstable systems
- AB09FD Balance and Truncate model reduction of coprime factors
- AB09GD Singular perturbation approximation of coprime factors
- AB09MD Balance and Truncate model reduction for the stable part
- AB09ND Singular perturbation approximation based model reduction for the stable part

State-Space Transformations

- TB01ID Balancing a system matrix for a given triplet

14.9.5 MATRIX_X

Purpose: Convert a discrete dynamic system into an internally balanced dynamic form.

Syntax: [SB, SIGMASQ, T]=DBALANCE (SD, NS)

Purpose: Compute a reduced order form of a discrete-time system.

Syntax: [SR, NSR]=DMREDUCE (SD, NS, KEEP)

Purpose: Compute a reduced order form of a continuous system.

Syntax: [SR, NSR]=MREDUCE (S, NS, KEEP)

Purpose: Perform model structure determination

Syntax: [THETA, COR, COV] =MSD (X, Y)

The other software packages dealing with model reduction include:

1. **MATRIX_X Model Reduction Module** (1998) by B. D. O. Anderson and B. James.
2. **μ -Analysis and Synthesis Toolbox 1.0** by G. Balas, J. Doyle, K. Glover, A. Packard and R. Smith (1998).
3. **Robust Control Toolbox** 2.0 by R. Y. Chiang and M. G. Safonov.

14.10 Summary and Review

The chapter covers the topics:

- Internal Balancing
 - Model Reduction
- I. **Internal Balancing.** Given an $n \times n$ stable system (A, B, C) , there always exists a transformation T that simultaneously diagonalizes both the controllability and observability Grammians to the same diagonal matrix $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$, where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} > \sigma_{r+2} > \dots > \sigma_n$. The numbers $\sigma_i, i = 1, \dots, n$ are the **Hankel singular values**.
- In this case, the transformed system $(\tilde{A}, \tilde{B}, \tilde{C})$, is called **internally balanced**. **Algorithm 14.2.1** and **Algorithm 14.2.2** compute, respectively, the internal balancing of a stable minimal and nonminimal realization of a **continuous time system**. The internal balancing of a **discrete-time system** is discussed in **Section 14.3**.

II. Model Reduction. The problem of model reduction is the problem of constructing a q -th order model from a given n -th order model ($n > q$) in such a way that the reduced q -th order model is close to the original system in some sense. The precise mathematical definition of model reduction appears in **Section 14.4**. Two methods for model reduction are described in this chapter.

Model Reduction via Internal Balancing

Once a system is internally balanced, a desired reduced order model can be obtained by truncating the states corresponding to the less controllable and observable modes (**Algorithm 14.4.1**).

Theorem 14.4.1 shows that a truncated model of order q is also balanced and stable, and furthermore, if $G(s)$ and $G_R(s)$ are the respective transfer functions of the original and the truncated model, then

$$\| G(s) - G_R(s) \|_{\infty} \leq 2(\sigma_{d+1}, \dots, \sigma_n),$$

where $\sigma_i, i = 1, \dots, n$ are the Hankel singular values and q is such that the states corresponding to $\sigma_{d+1}, \dots, \sigma_n$ are eliminated. This result thus gives one a criterion for choosing the optimal order q of the reduced-order model.

The Schur Method for Model Reduction

There are some numerical difficulties associated with the procedure of finding a reduced order model via internal balancing using Algorithm 14.2.1 and Algorithm 14.2.2. The transforming matrix T in Algorithm 14.2.1 and the matrices L and Z in Algorithm 14.2.2 can be, in some cases, highly ill-conditioned. An alternative method (**Algorithm 14.4.2**) for model reduction based on the real Schur decomposition of the product of the controllability and observability Grammians, is described in **Section 14.4. The transforming matrix T in this case is orthogonal, and, therefore, well-conditioned**. The Schur method does not give an internally balanced system; however, the essential properties of the original system are preserved. In fact, **Theorem 14.4.2** shows that the d -th order transfer function matrix obtained by the Schur method is exactly the same as that of the one obtained via **Algorithm 14.2.1**.

A possible numerically difficulty with Algorithm 14.4.2 is the explicit computation of the product of the controllability and observability Grammians. In this case, instead of explicitly computing the controllability and observability Grammians, their Cholesky factors can be computed using the Hammarling algorithm (**Algorithm 8.6.1**) in Chapter 8. **This then leads to the square-root algorithm (Algorithm 14.2.2)**.

Combining the advantages of the Schur method and the square-root algorithm, a **balancing-free square root method** has been developed. This is described in **Section 14.4.3**.

Hankel-norm Approximation

Given a stable $G(s)$, the problem of finding a $\hat{G}(s)$ of McMillan degree k such that $\|G(s) - \hat{G}(s)\|_H$ is minimized is called an optimal Hankel-norm approximation.

A characterization of all solutions to Hankel-norm approximation is given in Section 14.5.1 (**Theorem 14.5.2**). An algorithm (**Algorithm 14.5.1**) for computing an optimal Hankel-norm approximation is then presented.

III. Model Reduction of an Unstable System.

The model reduction of an unstable system can be achieved by decomposing the model into its stable and unstable part, followed by finding a model reduction of the stable part and finally adding the reduced-order model of the stable part with the unstable part. This is described in Section 14.6. For this and another approach, based on stable rational coprime factorization, see Varga (2001).

IV. Weighted Model Reduction

Sometimes the errors at high frequencies in a reduced-order model can be reduced using weights on the model. This is discussed in Section 14.7.

V. Comparison of the Model Reduction Procedures

The model reduction procedures are summarized and a brief comparative discussion of different procedures is presented in **Section 14.8**.

14.11 Chapter Notes and Further Reading

The internal balancing algorithms, Algorithm 14.2.1 and Algorithm 14.2.2 are due to Laub (1980) and Tombs and Postlethwaite (1987), respectively. The idea of model reduction via balanced truncation was first introduced by Moore (1981).

The stability property of the truncated subsystem (part (a) of Theorem 14.4.1) was obtained by Pernebo and Silverman (1982) and the error bound (part (b) of Theorem 14.4.1) is due to Glover (1984) and Enns (1984).

The result was first obtained by Enns (1984) and Glover (1984) gave an independent proof. The Schur algorithm for model reduction and Theorem 14.4.2 is due to Safonov and Chiang (1989). The balancing-free square-root method for model reduction is due to Varga (1991).

The Hankel-norm approximation problem was introduced and solved by Glover in a celebrated paper (Glover (1984)). Besides the topic of Hankel-norm approximation of a transfer function, the paper contains many other beautiful results on systems theory and linear algebra. A good discussion of this topic can also be found in the book by Zhou, Doyle, and Glover (1996). See also Glover (1989).

For results on discrete-time balanced model reduction, see Al-Saggaf and Franklin (1987), Hinrichsen and Pritchard (1990).

The idea of frequency weighted model reduction is due to Enns (1984). Other subsequent results on this and related topics can be found in Al-Saggaf and Franklin (1988), Glover (1986, 1989),

Glover, Limebeer, and Hung (1992), Hung and Glover (1986), Liu and Anderson (1990), Zhou (1993), etc.

For a discussion on Balanced Stochastic Truncation (BST) method, see Zhou, Doyle and Glover (1996).

The idea of singular perturbation approximation is due to Liu and Anderson (1989).

A recent book by Obinata and Anderson (2000) deals exclusively with the topic of model reduction.

The paper by Green (1988) deals with stochastic balanced realization. For more on this topic, see Zhou et al. (1996).

EXERCISES

1. Prove part (a) of Theorem 14.4.1 and fill in the missing details of part (b), whenever indicated in the book.
2. Let $G(s) = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$. Suppose that there exists a symmetric matrix $P = \text{diag}(P_1, 0)$, with P_1 nonsingular, such that

$$AP + PA^T + BB^T = 0.$$

Partition $G(s)$ conformably with P as

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} \hat{A}_R & A_{12} & \hat{B}_R \\ A_{21} & A_{22} & B_2 \\ \hat{C}_R & C_2 & D \end{bmatrix}$$

Then prove that $\begin{bmatrix} \hat{A}_R & \hat{B}_R \\ \hat{C}_R & D \end{bmatrix}$ is also a realization of $G(s)$. Moreover, if \hat{A}_R is stable, then (\hat{A}_R, \hat{B}_R) is controllable.

3. Based on the result of Exercise 2 develop a method for extracting a controllable subsystem from a stable noncontrollable system.
4. Let $G(s)$ be the same as in Exercise 2. Suppose that there exists a symmetric matrix $Q = \text{diag}(Q_1, 0)$, with Q_1 is nonsingular, such that $QA + A^TQ + C^TC = 0$. Partition the realization (A, B, C, D) conformably with Q as in Exercise 2. Then prove that $\begin{bmatrix} \hat{A}_R & \hat{B}_R \\ \hat{C}_R & D \end{bmatrix}$ is also a realization of $G(s)$. Prove further that (\hat{A}_R, \hat{C}_R) is observable if \hat{A}_R is stable.
5. Based on Exercise 4, develop a method for extracting an observable subsystem from a stable nonobservable system.
6. Construct your own example to illustrate the numerical difficulties of Algorithm 14.2.1.
7. Prove that the rows $\{1, \dots, d\}$ and the rows $\{d+1, \dots, n\}$ of T^{-1} in Algorithm 14.2.1, form bases for the left eigenspaces of the matrix $C_G O_G$ associated with the eigenvalues $\{\sigma_1^2, \dots, \sigma_d^2\}$, and $\{\sigma_{d+1}^2, \dots, \sigma_n^2\}$, respectively.
8. Prove that the columns of the matrix V_{1S} and those of the matrix U_{2S} in the Schur algorithm (Algorithm 14.4.2) for model reduction, form orthonormal bases for the right and left invariant subspace of $C_G O_G$ associated with the large eigenvalues $\sigma_1^2, \dots, \sigma_d^2$.

9. Prove that the controllability and observability Grammians of the reduced-order model obtained by the Schur algorithm (Algorithm 14.4.2) are, respectively, given by $\hat{C}_G^R = S_1^T C_G S_1$ and $\hat{O}_G^R = S_2 O_G S_2$, where C_G and O_G are the controllability and observability Grammians of the original model.
10. (a) Modify the Schur algorithm for model reduction by making use of Hammarling's algorithm (**Algorithm 8.6.1**) so that the explicit formation of the product $C_G O_G$ is avoided, and only the Cholesky factors L_c and L_o are computed.
 (b) Work out an example to demonstrate the superiority of this modified Schur algorithm over the Schur algorithm.
11. (a) Prove that the matrix T defined by (14.3.4) transforms the discrete-time system (14.3.1) to the balanced system (14.3.5).
 (b) Write down an algorithm for the internal balancing of a discrete-time system
 (c) Develop a discrete-analogue of Algorithm 14.2.2.

12. Let $G(s) = \begin{bmatrix} A & B \\ C & O \end{bmatrix}$ be the transfer function of a balanced realization. Then prove that

$$\sigma_1 \leq \| G \|_{\infty} \leq \int_0^{\infty} \| Ce^{At} B \| dt \leq 2 \sum_{i=1}^n \sigma_i,$$

where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n) \geq 0$ is the controllability and the observability Grammian of the realization and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$.

13. Prove that if the diagonal entries of the internal balancing algorithm Σ are all distinct, then every subsystem of the balanced system is asymptotically stable.
14. Construct an example to show that the bound of Theorem 14.4.1 can be loose if the quantities $\sigma_i, i = 1, \dots, n$ are close to each other.

Hint: Construct a stable realization $G(s)$ such that $G^T(-s)G(s) = I$ and then construct a balanced realization of $G(s)$. Now make a small perturbation to this balanced realization and work with this perturbed system.

15. (a) Develop a Schur-method for model reduction of the discrete-time system.
 (b) Give a simple example to illustrate the method.
 (c) Give a flop-count of the method.
16. (**Minimal Realization Using Block Diagonalization**) (Varga (1991)). Consider the following algorithm:

Step 1. Reduce A to block diagonal form and update B and C ; that is find a nonsingular matrix T such that $T^{-1}AT = \text{diag}(\bar{A}_1, \bar{A}_2, \dots, \bar{A}_r)$, $T^{-1}B = (\bar{B}_1, \bar{B}_2, \dots, \bar{B}_r)$, $CT = (\bar{C}_1, \bar{C}_2, \dots, \bar{C}_r)$.

Assume that the diagonal blocks in A have disjoint spectra.

Step 2. Find a minimal realization of each of the system $(\bar{A}_i, \bar{B}_i, \bar{C}_i), i = 1, \dots, r$ using Algorithm 14.2.1 or 14.2.2, as appropriate.

Let $(\hat{A}_i, \hat{B}_i, \hat{C}_i), i = 1, \dots, r$ be the computed minimal realization of $(\bar{A}_i, \bar{B}_i, \bar{C}_i)$ in step 2.

Then show that the system $(\hat{A}, \hat{B}, \hat{C})$ defined by:

$$\hat{A} = \text{diag}(\hat{A}_1, \hat{A}_2, \dots, \hat{A}_r), \quad \hat{B} = \begin{pmatrix} \hat{B}_1 \\ \hat{B}_2 \\ \vdots \\ \hat{B}_r \end{pmatrix}, \quad \hat{C} = (\hat{C}_1, \hat{C}_2, \dots, \hat{C}_r),$$

is a minimal realization of (A, B, C) .

17. Using the matrix version of bilinear transformation

$$s = \frac{z-1}{z+1}$$

prove that the Hankel singular values of the discrete and continuous systems are identical.

(Hint: Obtain the system matrices (A, B, C) for the continuous-time system from the system matrices $(\bar{A}, \bar{B}, \bar{C})$ of the discrete-time system and then show that the controllability Grammian (observability Grammian) of (A, B, C) is the same as the controllability Grammian (observability Grammian) of $(\bar{A}, \bar{B}, \bar{C})$).

18. Using the bilinear transformation of Exercise 17 and Algorithm 14.5.1, find an optimal Hankel-norm approximation for the discrete-time system defined by the matrices in Example 14.3.1.

19. Write down a discrete analogue of Algorithm 14.2.2 and apply the algorithm to the system (A, B, C) defined by $A = \begin{pmatrix} 0.0001 & 1 & 0 \\ 0 & 0.1200 & 1 \\ 0 & 0 & 0 \end{pmatrix}$, $B = (1, 1, 0)^T$, $C = (1, 1, 1)$.

20. (S. S. Routh, 1929) Consider the system (A, B, C) given by

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 10^{-3} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 10^{-3} \end{pmatrix}^T$$

$$C = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 5 \times 10^5 \\ 0 & 0 & 0 & 0 & 0 & 0 & -6 & 1 & -2 & 5 \times 10^5 \end{pmatrix}$$

Find a reduced-order model of order 4 using

- (a) Balanced truncation via Algorithm 14.2.1 and 14.2.2.
- (b) The Schur method (Algorithm 14.4.2).

Compare the results with respect to the condition numbers of the transforming matrices and the $\|\cdot\|_\infty$ norm errors.

21. Prove that the weighting Grammians \bar{C}_G and \bar{O}_G are given by the equations (14.7.2) and (14.7.3).
22. Consider the two special cases of the frequency weighted model reduction:

Case 1. $W_i(s) = I$ and $W_o(s) \neq I$

Case 2. $W_i(s) \neq I$ and $W_o(s) = I$.

Prove that the reduced-order model $(\bar{A}_R, \bar{B}_R, \bar{C}_R)$ is stable provided that it is controllable in Case 1 and is observable in Case 2.

(Hint: Write the balanced Grammian $\Sigma = \text{diag}(\Sigma_1, \Sigma_2)$. Then show that

$$\bar{A}_R \Sigma_1 + \Sigma_1 \bar{A}_R^T + \bar{B}_R \bar{B}_R^T = 0$$

and

$$\bar{A}_R^T \Sigma_1 + \Sigma_1 \bar{A}_R + \bar{C}_R^T \bar{C}_R = 0).$$

Work out an example to illustrate the result.

23. (Singular Perturbation Approximations)

Let $(\bar{A}, \bar{B}, \bar{C})$ be a balanced realization of (A, B, C) . Partition the matrices $\bar{A}, \bar{B}, \bar{C}$ as:

$$\bar{A} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} \bar{B}_1 \\ \bar{B}_2 \end{pmatrix}, \quad \bar{C} = (\bar{C}_1, \bar{C}_2)$$

Then the system $(\hat{A}, \hat{B}, \hat{C})$ defined by

$$\begin{aligned} \hat{A} &= \bar{A}_{11} + \bar{A}_{12}(\gamma I - \bar{A}_{22})^{-1} \bar{A}_{21} \\ \hat{B} &= \bar{B}_1 + \bar{A}_{12}(\gamma I - \bar{A}_{22})^{-1} \bar{B}_2 \\ \hat{C} &= \bar{C}_1 + \bar{C}_2(\gamma I - \bar{A}_{22})^{-1} \bar{A}_{21} \end{aligned}$$

is called **balanced singular perturbation approximation** of $(\bar{A}, \bar{B}, \bar{C})$ (**Liu and Anderson (1989)**). ($\gamma = 0$ for a continuous-time system and $\gamma = 1$ for a discrete-time system).

Compute singular perturbation approximations of the system in Example 14.2.1

using Algorithm 14.2.1 and Algorithm 14.2.2.

(b) Show how the balancing-free square root method in Section 14.4.3 can be modified to compute singular perturbation approximation (**Hint.** Find the SVD of $Y^T X$ and then compute L and Z from the matrices of the SVD). See Varga (1991)

(c) Apply the modified balancing-free square-root method in (b) to the system in Example 14.2.1 and compare the results.

References

1. U.M. Al-Saggaf and G.F. Franklin, An error bound for a discrete reduced order model of a linear multivariable system, *IEEE Trans. Automat. Contr.*, vol. AC-32, pp. 815-819, 1987.
2. U.M. Al-Saggaf and G.F. Franklin, Model reduction via balanced realizations: an extension and frequency weighting techniques, *IEEE Trans. Automat. Contr.*, vol. AC-33, no. 7, pp. 687-692, 1988.
3. G. Balas, J. Doyle, K. Glover, A. Packard, and R. Smith, *μ -Analysis and Synthesis Toolbox* 3.0.4, The MathWorks Inc., Natick, MA, 1998.
4. R.Y. Chiang and M.G. Safonov, *Robust Control Toolbox 2.0.6.*, The MathWorks Inc., Natick, MA, 1997.
5. D.F. Enns, Model reduction with balanced realizations: An error bound and a frequency weighted generalization, *Proc. IEEE Conf. Dec. Control*, pp. 127-132, 1984.
6. K. Glover, Robust stabilization of linear multivariable systems: Relations to approximation, *Int. J. Contr.*, vol. 43, no. 3, pp. 741-766, 1986.
7. K. Glover, *A tutorial on Hankel-norm Approximation, Data to Model*, Springer-Verlag, New York, 1989. (J.C. Willems, Editor)
8. K. Glover, D.J.N. Limebeer, and Y.S. Hung, A structured approximation problem with applications to frequency weighted model reduction, *IEEE Trans. Automat. Control*, vol. AC-37, no. 4, pp. 447-465, 1992.
9. K. Glover, All optimal Hankel-norm approximations of linear multivariable systems and their L^∞ -error bounds, *Int. J. Control.*, 39:1115-1193, 1984.
10. M. Green, Balanced stochastic realizations, *Lin. Alg. Appl.*, vol. 98, pp. 211-247, 1988.
11. M.T. Heath, A.J. Laub, C.C. Paige, and R.C. Ward, Computing the singular value decomposition of the product of two matrices, *SIAM J. Sci. Stat. Comput.*, vol. 7, pp.1147-1159, 1986.
12. D. Hinrichsen and A.J. Pritchard, An improved error estimate for reduced-order models of discrete time system, *IEEE Trans. Automat. Control*, AC-35, pp. 317-320, 1990.
13. Y.S. Hung and K. Glover, Optimal Hankel-norm approximation of stable systems with first order stable weighting functions, *Syst. Contr. Lett.*, vol. 7, pp. 165-172, 1986.
14. T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.

15. S.K. Kung, and D.W. Lin, Optimal Hankel norm model reduction: multivariable systems, *IEEE Trans. Automat. Contr.*, vol. AC-26, pp. 832-852, 1981.
16. G.A. Latham and B.D.O. Anderson, Frequency weighted optimal Hankel-norm approximation of stable transfer function, *Syst. Contr. Lett.*, vol. 5 pp. 229-236, 1986.
17. A.J. Laub, Computation of ‘balancing’ transformations, *Proc. 1980 JACC*, Session FA8-E, 1980.
18. Y. Liu and B.D.O. Anderson, Singular perturbation approximation of balanced systems. *Int. J. Control*, vol. 50, pp. 1379-1405, 1989.
19. Y. Liu and B.D.O. Anderson, Frequency weighted controller reduction methods and loop transfer recovery, *Automatica*, vol. 26, no. 3, pp. 487-497, 1990.
20. MATLAB, *Control System Toolbox* 4.2, The MathWorks Inc., Natick, MA, 1998.
21. MATRIX_X, *Xmath Model Reduction module*, ISI, Santa Clara, CA, January 1998.
22. B.C. Moore, Principal component analysis in linear systems: Controllability, observability and model reduction, *IEEE Trans. Auto. Control*, AC-21, pp. 689-692, 1981.
23. G. Obinata and B.D.O. Anderson, *Model Reduction for Control System Design*, Springer-Verlag, New York, 2000.
24. L. Pernebo, and L.M. Silverman, Model reduction via balanced state space representation, *IEEE Trans. Automat. Contr.*, vol. AC-27, no. 2, pp. 382-387, 1982.
25. M.G. Safonov and R.Y. Chiang, A Schur method for balanced-truncation model reduction, *IEEE Trans. Automat. Control*, vol. 34, pp. 729-733, 1989.
26. M.S. Tombs and I. Postlethwaite, Truncated balanced realization of a stable non-minimal state-space system, *Int. J. Control*, vol. 46, pp. 1319-1330, 1987.
27. A. Varga, Balancing-free square-root algorithm for computing singular perturbation approximations, *Proc. 30th IEEE Conf. Dec. Control*, pp. 1062-1065, 1991.
28. A. Varga, Numerical methods and software tools for model reduction, *Proc. 1st MATHMOD Conf.*, Vienna, vol. 2, pp. 226-230, 1994 (I. Troch and F. Breitenecker, Editors).
29. A. Varga and V. Ionescu, HTOOLS-A Toolbox for solving H_∞ and H_2 synthesis problems, *Proc. IFAC/IMACS Symp. Computer Aided Design of Control Systems*, pp. 508-511, 1991.
30. A. Varga, Model Reduction Software in the SLICOT Library, *Applied and Computational Control, Signals and Circuits: Recent Developments*, Kluwer Academic publisher, Boston, pp. 239-282, 2001 (B.N. Datta, et al., Editors).

31. K. Zhou, J.C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice Hall, Upper Saddle River, NJ, 1996.
32. K. Zhou, Frequency weighted model reduction with \mathcal{L}_∞ error bounds, *Syst. Contr. Lett.*, vol. 21, pp. 115-125, 1993.

PART IV

SPECIAL TOPICS

Chapter 15. Large-Scale Matrix Computations in Control: Krylov Subspace Methods

Chapter 16. Some Existing Software for Control Systems Design and Analysis

Chapter 15

LARGE-SCALE MATRIX COMPUTATIONS IN CONTROL: KRYLOV SUBSPACE METHODS

Contents

15.1 Introduction	810
15.2 A General Discussion on Krylov Subspace Methods	811
15.3 The Arnoldi and GMRES Methods	813
15.3.1 The Arnoldi Method	813
15.3.2 Solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ using the Arnoldi Method	815
15.3.3 The GMRES Method for Solving $\mathbf{A}\mathbf{x} = \mathbf{b}$	820
15.3.4 Solving Shifted Linear Systems Using the Arnoldi Method	821
15.3.5 The Block Arnoldi Method	822
15.4 The Lanczos Method	823
15.4.1 The Block Lanczos Method.	825
15.5 The Krylov Subspace Methods in Control	826
15.5.1 Scopes of using the Krylov Subspace Methods in Control	826
15.5.2 The Controllability and the observability problems and the Krylov Subspace Methods	827
15.5.3 Arnoldi Method For Rank-one Lyapunov Equation	828
15.5.4 Restarted Arnoldi Method for Lyapunov Equation	833
15.5.5 Block Arnoldi Method For Discrete Lyapunov Equation	834
15.5.6 Arnoldi Methods for Sylvester Equation	838
15.5.7 Block Arnoldi Methods for Sylvester Equation	843
15.5.8 Arnoldi-Method for Sylvester-Observer Equation (single-output case) .	848
15.5.9 Arnoldi-Method for Continuous-time Algebraic Riccati Equation . .	854

15.5.10 Arnoldi Method for Partial Eigenvalue Assignment	858
15.5.11 Lanczos and Arnoldi Methods for Model Reduction	862
15.6 Summary and Review	879
15.7 Chapter Notes and Further Reading	883

Topics Covered

- Basic Krylov Subspace Methods (Lanczos and Arnoldi) and their Applications to Linear Systems Problems
- Krylov Subspace Methods for Controllability: Lyapunov, Sylvester, and Algebraic Riccati Equations; Partial Eigenvalue Assignment Problem and Model Reduction

15.1 Introduction

Several practical situations such as the design of Large Space Structures (LSS), computer networks, power systems, etc. give rise to large-scale computation in control. The computational complexity and storage requirements can grow exponentially with the dimensions of these problems.

Unfortunately, the numerical methods for control problems which have been described in previous chapters, are not suitable for large-scale solutions. Like most large practical problems, these problems are also very sparse. Very often the sparsity is an asset. It is conveniently exploited in the design of algorithms for solving large-scale problems. As we have seen before, the most numerically effective control algorithms, such as the Schur algorithm for the Lyapunov equation and the Hessenberg-Schur algorithm for the Sylvester equation (**Chapter 8**), the generalized Schur algorithm for the algebraic Riccati equations (**Chapter 13**), the observer-Hessenberg algorithm for the Sylvester-Observer equation (**Chapter 12**) and the Schur algorithm for model reduction (**Chapter 14**), etc., are not capable of preserving the inherited sparsity. This is because, the fundamental numerical linear algebra techniques such as the *Householder method for Hessenberg decomposition*, the *QR iteration for real-Schur decomposition*, *Gaussian elimination process for solving linear systems etc.*, which form bases of the above algorithms, are well-known to destroy the sparsity. Furthermore, for problems of size n , these are all $O(n^3)$ methods, and thus, are computationally prohibitive for large problems.

On the other hand, there have been some fine recent developments in the area of large-scale matrix computations. A class of classical methods known as the **Krylov subspace methods** have been found to be suitable for sparse matrix computations. The reason is that these methods can be implemented using matrix-vector multiplications only; therefore, the sparsity in the original problems can be preserved. The examples are the GMRES (Generalized Minimal

Residual) and the QMR (Quasi-Minimal Residual) methods for linear systems problem; the Arnoldi, Lanczos, and the Jacobi-Davidson methods, and several variants such as the restarted and block Arnoldi methods and band Lanczos method for eigenvalue problems.

It is only natural to develop algorithms for large-scale control problems using these effective large-scale techniques of matrix computations. Some work to this effect has been done in the last few years.

In this chapter, we will briefly review some of these methods.

These include:

- (i) Arnoldi Methods for the Lyapunov equation (**Algorithms 15.5.1-15.5.3**); the Sylvester equation (**Algorithm 15.5.4** and **15.5.5**); the Sylvester-observer equation (**Algorithm 15.5.6**), the continuous-time algebraic Riccati equation (**Algorithm 15.5.7**).
- (ii) Arnoldi method for multi-input partial eigenvalue assignment problem (**Algorithm 15.5.8**).
- (iii) Lanczos and Arnoldi methods for model reduction (**Algorithms 15.5.9-15.5.11**).

The organization of this chapter is as follows:

In **Section 15.2**, we give a general discussion on Krylov subspace methods and establish some notations.

In **Sections 15.3 and 15.4**, we state the basic Arnoldi and Lanczos methods and show how the Arnoldi methods can be applied to solve large and sparse linear systems $Ax = b$.

Section 15.5 is the main section of this chapter. This section describes Arnoldi and Lanczos methods for controllability and observability problems; Lyapunov, Sylvester, Sylvester-observer and Riccati equations; partial eigenvalue assignment and model reduction problems.

We stress that *the descriptions of our Krylov Subspace Methods given in Sections 15.3 and 15.4 are basic*. For practically implementable versions of these methods and associated software, we refer the readers to the books by Dongarra et al. (1998), Bai et al. (2000), Barrett et al. (1994) and Lehouce et al. (1988). *In particular, the homepage, ETHOME of the book by Bai et al. (2000) contains valuable information of available software.* Our only goal of this Chapter is to show the readers how these modern iterative numerical methods can be gainfully employed to solve some of the large and sparse control problems.

15.2 A General Discussion on Krylov Subspace Methods

The main purpose of this section is to establish some definitions and notations and discuss in the most general sense the two Krylov subspace methods: the **Lanczos** and the **Arnoldi methods**.

Given an $n \times n$ matrix A , and an n -vector x , the sequence $\{x, Ax, \dots, A^{n-1}x\}$ is called a **Krylov sequence**.

The subspace $K_m(A, x) = \text{span}\{x, Ax, \dots, A^{m-1}x\}$ is called the **Krylov subspace** of dimension m , assuming that the vectors are independent. A **block Krylov sequence** $\{B, AB, \dots, A^{n-1}B\}$

is generated by an $n \times n$ matrix A and an $n \times p$ matrix B , and

$$K_m(A, B) = \text{span}\{B, AB, \dots, A^{m-1}B\}$$

is the block **Krylov subspace** of dimension at most mp .

The methods that generate bases of the Krylov subspaces are called **Krylov subspace methods**. Among several Krylov subspace methods known in the literature, we will describe here only two: the **Lanczos** and **Arnoldi** Methods. Historically, the Lanczos method was the first of its type developed by C. Lanczos (Lanczos (1950)).

Starting with an $n \times n$ matrix A , and two n -vectors v and w , the **Lanczos method** constructs, after m steps, a set of **biorthogonal** vectors $\{v_1, \dots, v_m\}$; and $\{w_1, \dots, w_m\}$; and an $m \times m$ **tridiagonal matrix** T_m such that if $V_m = (v_1, \dots, v_m)$ and $W_m = (w_1, \dots, w_m)$, then $W_m^T A V_m = T_m$. The vectors $\{v_1, \dots, v_m\}$ form a basis of $K_m(A, v_1)$, and the vectors $\{w_1, \dots, w_m\}$ form a basis of $K_m(A^T, w_1)$.

Similarly, the **Arnoldi method**, originally introduced by Arnoldi (1951) in the context of solving matrix eigenvalue problems, constructs after m steps, starting with an $n \times n$ matrix A and a unit vector v , an $m \times m$ Hessenberg matrix H_m and an $n \times m$ orthonormal matrix $V_m = (v_1, \dots, v_m)$ such that $V_m^T A V_m = H_m$. The vectors $\{v_1, \dots, v_m\}$ form an orthonormal basis of $K_m(A, v_1)$. The **block Arnoldi method** is just a block version of the Arnoldi method which works with an $n \times n$ matrix and a set of starting vectors.

When A is a symmetric matrix, the Arnoldi method and the Lanczos method become identical, if $v_1 = w_1$.

The Arnoldi and Lanczos methods are used as **projection methods** in numerical linear algebra to solve **large-scale** matrix computational problems which are typically sparse (see Bai, et al, (2000), Saad (1992a, 1996), Stewart (2001)). The matrix T_m is the projection of A obtained from an oblique projection process onto $K_m(A, v_1)$ and orthogonal to $K_m(A^T, w_1)$. Similarly, H_m is the projection of A onto $K_m(A, v_1)$ with respect to the orthogonal basis V_m .

A large problem is typically projected onto a Krylov subspace of dimension m ($m \ll n$), then the projected problem is solved using a standard technique, and finally an appropriate solution of the original problem is retrieved from the solution of the smaller projected problem. Thus, once H_m and T_m are constructed, some of their eigenvalues will provide approximations to certain eigenvalues of A . **The quality of approximation improves as m increases.** Similar remarks hold with the linear systems problem: $Ax = b$. We will discuss here in some details of the solutions of the linear system problem $Ax = b$ using the Arnoldi method.

There are some drawbacks with these methods. Both the methods might encounter “**breakdowns**” or “**near breakdowns**” during the process of orthogonalization. In case of the Arnoldi method, such a “breakdown” is considered to be “**happy breakdown**” for the eigenvalue problem; because, in such a case one obtains an exact eigenvalue. In case of the Lanczos method, this is, however, not always true.

The researchers in recent years have found out some cures of the “breakdowns” or “near breakdowns” with the Lanczos method. One such cure is based on the “**look-ahead**” idea.

A “look-ahead” Lanczos algorithm continues to the next step, even if there is a “breakdown” in the previous step. Once this “next step” is successful, the process is continued until another “breakdown” occurs (see Freund, Gutknecht and Nachtigal (1993), Parlett, Taylor, and Liu (1985)).

Another concern with these methods is the **gradual loss of orthogonality** of the generated vectors. To cure these problems, some forms of “partial” and “selective” **reorthogonalizations** have been suggested. There are some practical difficulties again with these processes. For details, we refer the readers to the books of Saad (1992a), (1996) and Bai et al. (2000). To conclude this section, **let us mention that the research in this area of matrix computations is active and dynamic.**

15.3 The Arnoldi and GMRES Methods

In this section, we summarize the essentials of the Arnoldi method and its applications to solve the system $Ax = b$.

15.3.1 The Arnoldi Method

Given an $n \times n$ matrix A , a vector v , and an integer $m \leq n$, the Arnoldi method computes simultaneously a set of orthonormal vectors $\{v_1, \dots, v_m\}$, and an $m \times m$ Hessenberg matrix H_m such that

$$V_m^T A V_m = H_m,$$

where $V_m = (v_1, v_2, \dots, v_m)$.

The vectors $\{v_1, \dots, v_m\}$ form an orthonormal basis of the Krylov subspace $K_m(A, v_1) = \text{span}\{v_1, Av_1, \dots, A^{m-1}v_1\}$.

The development of the algorithm is conceptually simple. Suppose that $v_1 = \frac{v}{\|v\|_2}$ and let h_{11} be computed as $h_{11} = v_1^T A v_1$. Then the vector $\hat{v} = Av_1 - h_{11}v_1$ is orthogonal to v_1 . Thus, we can take $v_2 = \frac{\hat{v}}{\|\hat{v}\|_2}$ and $\|\hat{v}\|_2 = h_{21}$; so that $\{v_1, v_2\}$ form an orthonormal set.

The process can be continued. At step k , the entries of the k^{th} column of the matrix H together with v_k can be generated exploiting the fact that $\{v_1, v_2, \dots, v_k\}$ are orthonormal.

Algorithmically, $h_{k,k}$ will not be computed from the relation $h_{k,k} = v_k^T A v_k$ (see **Algorithm 15.3.1**).

There are several implementations of the Arnoldi method: **The classical Gram-Schmidt**, the **Modified Gram-Schmidt**, and the **Householder versions**, etc. We shall describe the **modified Gram-Schmidt implementation** here, which is known to be numerically more effective than the classical Gram-Schmidt version. For details of other versions of the Arnoldi method, see Saad (1996). The Householder Arnoldi was developed by Walker (1988).

Algorithm 15.3.1 The Arnoldi Method (The Modified Gram-Schmidt Version)

Inputs: A - An $n \times n$ matrix

v - An n -vector

m - An integer less than or equal to n .

Outputs: $V_m = (v_1, \dots, v_m)$, an $n \times m$ orthonormal matrix.

H_m - An $m \times m$ Hessenberg matrix.

Results:

$$(i) V_m^T A V_m = H_m$$

(ii) The vectors $\{v_1, \dots, v_m\}$ form an orthonormal basis of the Krylov subspace $K_m(A, v_1) = \text{span } \{v_1, Av_1, \dots, A^{m-1}v_1\}$.

Step 0. Normalize the vector v to obtain v_1 :

$$v_1 = \frac{v}{\|v\|_2}.$$

Step 1. For $k = 1, 2, \dots, m$ do

$$\hat{v} = Av_k$$

For $j = 1, 2, \dots, k$ do

$$h_{j,k} = v_j^T \hat{v}$$

$$\hat{v} = \hat{v} - h_{j,k} v_j$$

End

$$h_{k+1,k} = \|\hat{v}\|_2$$

$$v_{k+1} = \hat{v} / h_{k+1,k}$$

End

Define \tilde{H}_m as the $(m+1) \times m$ matrix whose nonzero entries are the coefficients h_{ij} , and H_m as the $m \times m$ matrix obtained from \tilde{H}_m by deleting its last row; let V_m be the $n \times m$ matrix whose j th column is the column vector v_j , i.e., $V_m \equiv (v_1, v_2, \dots, v_m)$. Then it is clear from the algorithm that we have the important equality:

$$AV_m = V_{m+1} \tilde{H}_m,$$

where

$$V_{m+1} = (v_1, v_2, \dots, v_{m+1}). \quad (15.3.1)$$

Equivalently,

$$AV_m - V_m H_m = h_{m+1,m}(0, 0, \dots, 0, v_{m+1}), \quad (15.3.2)$$

or

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T. \quad (15.3.3)$$

It is easy to establish that each of the vectors v_i is of the form

$$v_i = p_{i-1}(A)v_1,$$

where p_{i-1} is a polynomial of degree $i - 1$. Moreover, **the algorithm breaks down at step j , i.e., $\hat{v}_{j+1} = 0$, if and only if the degree of the minimal polynomial of v_1 is exactly j** , that is, it is a combination of j eigenvectors. As indicated earlier, this is a **happy breakdown** for the eigenvalue problem. In this case, the subspace $K_j(A, v_1)$ is invariant and the approximate eigenvalues and eigenvectors are exact.

The equation

$$AV_m - V_m H_m = h_{m+1,m} (0, 0, \dots, 0, v_{m+1}) \quad (15.3.4)$$

is referred to as the **Arnoldi equation**.

The vector v_1 is referred to as the **Arnoldi vector**.

The orthonormal basis formed by the vectors $\{v_1, \dots, v_m\}$ is called the **Arnoldi basis**.

Restarted Arnoldi Methods

The storage and computational costs of the Arnoldi method increase substantially as m increases (approximately $O(m^2n)$ flops and $(mn + \frac{m^2}{2})n$ storage locations). To overcome the difficulty, the Arnoldi method is usually restarted with a different starting vector (keeping m fixed) or by changing m dynamically introducing a fixed variable m_1 (a small integer) such that the accuracy of the method is checked every m_1 iterations of the Arnoldi method. Such variations of the Arnoldi method are usually called the **Restarted Arnoldi methods** and will be described in the sequel, in the context of the applications of the Arnoldi methods to solving linear systems and control problems. There are, however, other types of restarted Arnoldi methods. We will discuss some such restarted Arnoldi methods in Section 15.5.11.

15.3.2 Solving $Ax = b$ using the Arnoldi Method

The Arnoldi method, described in the previous section, can be conveniently used to solve the large and sparse linear system

$$Ax = b.$$

We will describe two methods: A **Galerkin method** and a **minimal residual method**. The basic idea behind both the methods is the same: both are **projection methods**. First, the original problem of dimension n is projected into a smaller problem of dimension m by constructing an orthonormal basis of the associated Krylov subspace using the Arnoldi method, the projected smaller problem is then solved using a standard technique, and then finally an approximation of the solution of the original problem is computed from the solution of the projected problem. The two methods differ in the way the projected problem is solved. A common outline of the methods is as follows.

First, an initial approximate x_0 of the solution vector x is guessed, then the corresponding residual vector $r_0 = b - Ax_0$ is computed. A correction vector z_0 is then obtained such that

$$Ax_1 = A(x_0 + z_0) \cong b.$$

To determine z_0 , the m -steps of the Arnoldi method, starting with $v_1 = r_0/\|r_0\|_2$, can be run to generate the matrices H_m , V_m , and \tilde{H}_m , and then z_0 is sought in the form

$$z_0 = V_m y_0$$

for some $y_0 \in \mathbb{R}^m$.

The vector y_0 is computed differently by the two methods.

In the **Galerkin type of** method, it is required that the residual vector $r_1 = r_0 - AV_m y_0$ be orthogonal to $K_m(A, r_0)$. This gives

$$V_m^T(r_0 - AV_m y_0) = 0$$

That is,

$$V_m^T A V_m y_0 = V_m^T r_0$$

or

$$H_m y_0 = \|r_0\|_2 e_1.$$

This projected problem is now solved to obtain y_0 , then the correction vector $z_0 = V_m y_0$ is computed, and finally an improved solution vector $x_1 = x_0 + z_0$ is found.

In the generalized minimized residual method, a correction vector $z_0 = V_m y_0$ is sought such that the residual vector $r_1 = r_0 - Az_0$ is minimized.

If the corresponding residual vector $r_1 = b - Ax_1$, is not small enough, then the process is restarted setting $x_0 \equiv x_1$ and $r_0 \equiv r_1$. Other types of restarted methods, such as restarting with an increased value of m , are also possible. But, we will consider here restarted methods only of the first type.

We state below a **Restarted Arnoldi Method of Galerkin-type for solving $Ax = b$** .

Algorithm 15.3.2 An Explicitly Restarted Arnoldi Algorithm for $Ax = b$ (Galerkin-Type)

Inputs: A - An $n \times n$ matrix, large and sparse
 m - A positive integer less than n
 ϵ - Tolerance (> 0).
 x_0 - An initial approximation

Output: Approximate solutions x_0 such that the associated residual vectors $r_0 = b - Ax_0$ satisfy $\|r_0\|_2 < \epsilon$.

Step 1. Compute $r_0 = b - Ax_0$.

Step 2. If $\|r_0\|_2 < \epsilon$, stop.

Step 3. Run m steps of the Arnoldi algorithm to generate the matrices V_m and H_m using $v_1 = r_0 / \|r_0\|_2$.

Step 4. Solve the $m \times m$ system

$$H_m y_0 = \|r_0\|_2 e_1$$

Step 5. Compute the correction vector z_0

$$z_0 = V_m y_0.$$

Step 6. Compute the new solution vector x_1

$$x_1 = x_0 + z_0.$$

and set $x_0 \equiv x_1$.

Step 7. Go to Step 1.

Example 15.3.1 Consider the solution of $Ax = b$, with $\epsilon = 0.1$, $m = 2$,

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 7 \end{pmatrix}$$

$$b = \begin{pmatrix} 2 & 2 & 3 & 4 & 5 & 6 & 8 \end{pmatrix}^T$$

$$x_0 = \begin{pmatrix} 0.6000 & 0.6000 & 0.6000 & 0.6000 & 0.6000 & 0.6000 & 0.6000 \end{pmatrix}^T$$

$$\text{Step 1. } r_0 = \begin{pmatrix} 0.8000 & 0.8000 & 1.2000 & 1.6000 & 2 & 2.4000 & 3.2000 \end{pmatrix}^T.$$

Step 2: $|r_0|_2 = 5.027922 > 0.1$.

Step 3:

$$v_1 = \begin{pmatrix} 0.1591 & 0.1591 & 0.2387 & 0.3182 & 0.3978 & 0.4773 & 0.6364 \end{pmatrix}^T,$$

$$V_m = \begin{pmatrix} 0.1591 & -0.0928 \\ 0.1591 & -0.4209 \\ 0.2387 & -0.4673 \\ 0.3182 & -0.4043 \\ 0.3978 & -0.2319 \\ 0.4773 & 0.0498 \\ 0.6364 & 0.6133 \end{pmatrix}, \quad H_m = \begin{pmatrix} 5.8481 & 1.4547 \\ 1.4547 & 4.4751 \end{pmatrix}.$$

Step 4. $y_0 = \begin{pmatrix} 0.9354 & -0.3041 \end{pmatrix}^T$.

Step 5. $z_0 = \begin{pmatrix} 0.1770 & 0.2768 & 0.3653 & 0.4206 & 0.4426 & 0.4313 & 0.4088 \end{pmatrix}^T$

Step 6. $x_1 = \begin{pmatrix} 0.7770 & 0.8768 & 0.9653 & 1.0206 & 1.0426 & 1.0313 & 1.0088 \end{pmatrix}^T$

Step 7. *Restart with $x_0 \equiv x_1$.*

$$r_0 = \begin{pmatrix} 0.2141 & 0.2464 & 0.1040 & -0.0824 & -0.2130 & -0.1880 & 0.1612 \end{pmatrix}^T$$

$$|r_0|_2 = 0.480460 > 0.1.$$

$$v_1 = \begin{pmatrix} 0.4457 & 0.5128 & 0.2165 & -0.1714 & -0.4433 & -0.3913 & 0.3354 \end{pmatrix}^T$$

$$V_m = \begin{pmatrix} 0.4457 & -0.4413 \\ 0.5128 & -0.4511 \\ 0.2165 & -0.0938 \\ -0.1714 & -0.0022 \\ -0.4433 & -0.2037 \\ -0.3913 & -0.3545 \\ 0.3354 & 0.6526 \end{pmatrix}, \quad H_m = \begin{pmatrix} 3.9707 & 2.2402 \\ 2.2402 & 3.9945 \end{pmatrix}.$$

$$y_0 = \begin{pmatrix} 0.1770 & -0.0993 \end{pmatrix}^T$$

$$z_0 = \begin{pmatrix} 0.1227 & 0.1356 & 0.0476 & -0.0301 & -0.0582 & -0.0341 & -0.0054 \end{pmatrix}^T$$

$$x_2 = \begin{pmatrix} 0.8997 & 1.0124 & 1.0130 & 0.9905 & 0.9844 & 0.9973 & 1.0034 \end{pmatrix}^T$$

Step 7. *Restart with $x_0 \equiv x_2$.*

$$r = 10^{-1} \begin{pmatrix} 0.9684 & -0.2473 & -0.3888 & 0.3812 & 0.7825 & 0.1645 & 0.7629 \end{pmatrix}^T$$

$$|r_0|_2 = 0.158646 > 0.1.$$

$$v_1 = \begin{pmatrix} 0.6104 & -0.1559 & -0.2451 & 0.2403 & 0.4932 & 0.1037 & 0.4809 \end{pmatrix}^T$$

$$V_m = \begin{pmatrix} 0.6104 & -0.6117 \\ -0.1559 & 0.1431 \\ -0.2451 & 0.1280 \\ 0.2403 & -0.0304 \\ 0.4932 & 0.1329 \\ 0.1037 & 0.0690 \\ 0.4809 & 0.7520 \end{pmatrix}, \quad H_m = \begin{pmatrix} 4.3192 & 2.5260 \\ 2.5260 & 3.6239 \end{pmatrix}.$$

$$y_0 = 10^{-1} \begin{pmatrix} 0.6201 & -0.4322 \end{pmatrix}^T$$

$$z_0 = 10^{-1} \begin{pmatrix} 0.6429 & -0.1585 & -0.2073 & 0.1621 & 0.2484 & 0.0345 & -0.0268 \end{pmatrix}^T$$

$$x_3 = \begin{pmatrix} 0.9640 & 0.9965 & 0.9922 & 1.0067 & 1.0092 & 1.0007 & 1.0007 \end{pmatrix}^T$$

Step 7. *Restart with $x_0 \equiv x_3$.*

$$r = 10^{-1} \begin{pmatrix} 0.3524 & 0.0697 & 0.2331 & -0.2673 & -0.4594 & -0.0424 & 0.3080 \end{pmatrix}^T$$

Gives $|r_0|_2 = 0.0750 < 0.1$.

Example 15.3.2 (Numerical Experiment with Algorithm 15.3.2).

Algorithm 15.3.2 was run with a sparse matrix of order 99 and fixed $m = 4$. Using MATLAB notations, we write

$$A = \text{diag}((1 : 99)) + \text{diag}(1, 98) + \text{diag}(1, -98)$$

which is a sparse matrix with entries $1, \dots, 99$ on the diagonal and 1 on the top right and bottom left corners;

$$b = \begin{pmatrix} 2 & 2 & 3 & \dots & 97 & 98 & 100 \end{pmatrix}^T.$$

Choose the initial approximation $x_0 = \begin{pmatrix} 0.1 & 0.1 & \dots & \dots & 0.1 & 0.1 \end{pmatrix}^T$.

The norms of the residual vectors $b - Ax_i, i = 0, 1, \dots, 20$ are shown in the table below and plotted in Figure 15.1 using logarithmic scale. The plot shows that the restarted Arnoldi algorithm for $Ax = b$ converges as the number of iterations i increases.

Table 15.1

i	$ b - Ax_i _2$	i	$ b - Ax_i _2$	i	$ b - Ax_i _2$
0	515.875101	7	0.489985	14	0.055578
1	17.950331	8	0.337757	15	0.043757
2	4.739720	9	0.263640	16	0.030658
3	2.455344	10	0.183782	17	0.024151
4	1.380971	11	0.144290	18	0.016929
5	0.971805	12	0.100907	19	0.013341
6	0.641555	13	0.079375	20	0.009355

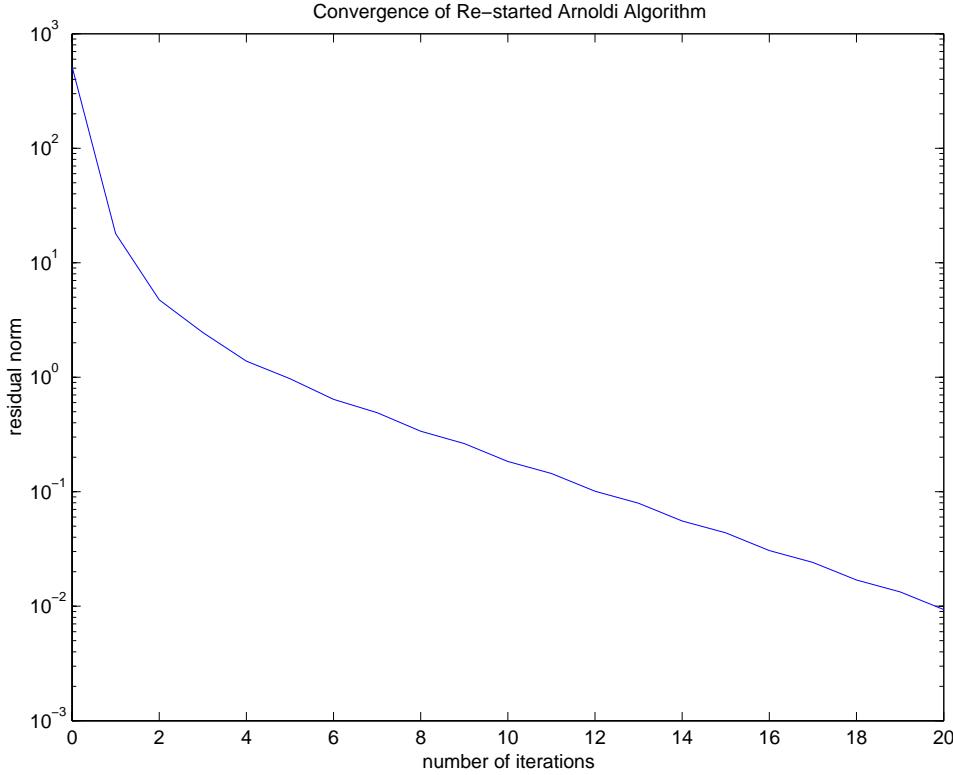


Figure 15.1: The Plot of the Norm of the Residual Vectors in Logarithmic Scale.

15.3.3 The GMRES Method for Solving $Ax = b$

We now present the other method, a minimized residual method, called the **Generalized Minimal Residual Method**, (GMRES), developed by Saad and Schultz (1986).

Here, one seeks a correction vector $z_0 = V_m y_0$ such that it minimizes the norm of the residual vector $r_1 = r_0 - Az_0$.

Since

$$\begin{aligned} r_1 &= r_0 - Az_0 = r_0 - AV_m y_0 = r_0 - V_{m+1} \tilde{H}_m y_0 \\ &= V_{m+1}(e_1 \|r_0\| - \tilde{H}_m y_0), \end{aligned}$$

we have

$$\begin{aligned} &\min_{z_0 \in K_m(A, r_0)} \|r_0 - Az_0\|_2 \\ &= \min_{y_0 \in \mathbb{R}^m} \|V_{m+1}(e_1 \|r_0\| - \tilde{H}_m y_0)\|_2 \\ &= \min_{y_0 \in \mathbb{R}^m} \|e_1 \|r_0\| - \tilde{H}_m y_0\|_2. \end{aligned}$$

This minimization problem can be solved using the QR factorization method described earlier (see Chapter 3). Once y_0 is so obtained, the correction vector $z_0 = V_m y_0$, the new improved solution vector $x_1 = x_0 + z_0$, and the corresponding residual vector $r_1 = b - Ax_1$, can easily be obtained.

If $\|r_1\|_2$ is not small enough, the process can be restarted by setting $x_0 \equiv x_1$ and $r_0 \equiv r_1$.

Algorithm 15.3.3 An Explicitly Restarted Generalized Minimal Residual Method (GMRES) for $Ax = b$

- Input:** A - An $n \times n$ large and sparse matrix
 x_0 - An initial approximation to the solution of $Ax = b$
 ϵ - Tolerance (> 0)
 m - A positive integer ($\leq n$).
- Output:** Approximations x_0 to the solutions of $Ax = b$ such that the associated residual vectors $r_0 = b - Ax_0$ satisfy $\|r_0\|_2 < \epsilon$.
- Step 1.** Compute $r_0 = b - Ax_0$.
- Step 2.** If $\|r_0\|_2 < \epsilon$, stop.
- Step 3.** Run m steps of the Arnoldi method with $v_1 = r_0/\|r_0\|_2$ to generate the $(m+1) \times m$ Hessenberg matrix \tilde{H}_m and the orthonormal matrix V_m .
- Step 4.** Find the vector y_0 such that the function $J(y) = \| \|r_0\|_2 e_1 - \tilde{H}_m y \|_2$ is minimized over all vector $y \in \mathbb{R}^m$; $e_1 = (1, 0, \dots, 0)^T \in \mathbb{R}^{n+1}$.
- Step 5.** Compute the correction vector
- $$z_0 = V_m y_0.$$
- Step 6.** Compute the new solution vector x_1 and set that to x_0 :
- $$x_0 \equiv x_1 = x_0 + z_0.$$
- Step 7.** Go to Step 1.

15.3.4 Solving Shifted Linear Systems Using the Arnoldi Method

An important observation is that the Arnoldi basis V_m is invariant under a diagonal shift σ of A : if we were to use $A - \sigma I$ instead of A , we would obtain the same sequence $\{v_1, \dots, v_m\}$. This is because the Krylov subspace $K_m(A, v_1)$ is the same as $K_m(A - \sigma I, v_1)$. Note that from (15.3.3) we have

$$(A - \sigma I)V_m = V_m(H_m - \sigma I) + h_{m+1,m}v_{m+1}e_m^T, \quad (15.3.5)$$

which means that if we run the Arnoldi method with the matrix $A - \sigma I$, we will obtain the same matrix V_m , but the matrix H_m will have its diagonal shifted by σI .

A consequence of this is that for solving several linear systems of the form

$$(A - \mu_i I)x_i = b, i = 1, 2, \dots,$$

we might use the same information V_m and H_m , generated only once, to solve these systems approximately. We will use this idea later in Section 15.5.8.

15.3.5 The Block Arnoldi Method

The Arnoldi method just described works with one single vector. But, in several applications, as we will see here, there is a need to work with a block of vectors rather than a single vector. There is a straight forward generalization of the Arnoldi algorithm that works with a block of vectors - it is called **the block Arnoldi algorithm**. The development of this algorithm follows along the same line as Algorithm 15.3.1; the only difference is that one now works with a block of vectors, rather than a single vector.

The block Arnoldi algorithm is particularly suited for handling multivariable control problems. We describe the algorithm in the following.

Algorithm 15.3.4 The Block Arnoldi Algorithm (Modified Gram-Schmidt Version)

Inputs: A - An $n \times n$ matrix

V - An $n \times p$ matrix with full rank

m - An integer less than n .

Outputs: $U_m = (V_1, \dots, V_m)$ - An orthonormal matrix

H_m = A block upper Hessenberg matrix of dimension mp .

Results: 1. $\{V_1, \dots, V_m\}$ form an orthonormal basis of the Krylov subspace

$K_m(A, V_1)$.

2. $U_m^T A U_m = H_m$ where $U_m = (V_1, \dots, V_m)$.

Step 0. Compute the $n \times p$ orthogonal matrix V_1 by finding the QR factorization of $V : V = V_1 R$. (Note that R is here $p \times p$).

Step 1. For $k = 1, 2, \dots, m$ do.

 Compute $\hat{V} = AV_k$.

 For $j = 1, 2, \dots, k$ do

$H_{j,k} = V_j^T \hat{V}$

$\hat{V} = \hat{V} - V_j H_{j,k}$

 End

 Compute $H_{k+1,k}$ by finding the QR factorization of $\hat{V} : \hat{V} = V_{k+1} H_{k+1,k}$

 End

The Block-Arnoldi Equation

Let $U_m = (V_1, V_2, \dots, V_m)$, $H_m = (H_{ij})$, $1 \leq i, j \leq m$ and $H_{ij} = 0$, $i > j + 1$. Then, it can be shown (Saad (1992a)) that the following analog of the Arnoldi equation holds in the block case:

$$AU_m - U_m H_m = V_{m+1} H_{m+1,m} E_m^T$$

where E_m is the matrix of the last p columns of the $mp \times mp$ identity matrix.

Also, note that

U_m is of order $n \times mp$

and

$$H_m = U_m^T A U_m.$$

Thus, H_m is a block upper Hessenberg matrix of order $mp \times mp$ and $U_m^T U_m = I_{mp \times mp}$.

Remarks:

1. In case V does not have full rank, we need to use the technique of QR factorization with column pivoting (see Datta (1995), Golub and Van Loan (1996)) to compute the QR factorization of V . The QR factorization of V with column pivoting (see **Chapter 3**) gives:

$$VP = \begin{bmatrix} \hat{V}_1, \hat{V}_2 \end{bmatrix} \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix}$$

in which R_1 is upper triangular whose rank is the same as that of V and P is a suitable permutation matrix. In such a case, we set

$$V_1 = \hat{V}_1 \text{ and } R = [R_1, R_2] P^T.$$

The QR factorization of \hat{V} in Step 1 can similarly be modified using the technique of column pivoting. We leave it to the readers as an exercise [**Exercise 1**] to work out the modified version of the Algorithm 15.3.4 incorporating the technique of column pivoting.

2. The block Arnoldi algorithm clearly breaks down if $H_{k+1,k}$ becomes zero for some k . Such a breakdown has positive consequences in some applications such as solutions of matrix equations (see **Section 15.5**).

15.4 The Lanczos Method

A classical well-known Krylov subspace algorithm, is the **Lanczos** algorithm due to C. Lanczos (1950). In fact, historically, the **symmetric Lanczos algorithm is the first of its kind**. For a nonsymmetric matrix A , the algorithm constructs, starting with two vectors v_1 , and w_1 , a **pair of biorthogonal bases** for the two Krylov subspaces:

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, \dots, A^{m-1}v_1\}$$

and

$$K_m(A^T, w_1) = \text{span}\{w_1, A^T w_1, \dots, (A^T)^{m-1} w_1\}.$$

Algorithm 15.4.1 *The Nonsymmetric Lanczos Algorithm*

Inputs: A - An $n \times n$ matrix
 v - An n -vector
 w - An n -vector

Output: $V_m = (v_1, \dots, v_m)$ - An $n \times m$ matrix
 $W_m = (w_1, \dots, w_m)$ - An $n \times m$ matrix
 T_m = An $m \times m$ **tridiagonal matrix**

Results: The set $\{v_1, \dots, v_m\}$ forms a basis of the Krylov subspace $K_m(A, v_1)$, and the set $\{w_1, \dots, w_m\}$ forms a basis of the Krylov subspace $K_m(A^T, w_1)$.

Step 0. Scale the vectors v and w to get the vectors v_1 and w_1 such that $w_1^T v_1 = 1$.
Set $\beta_1 \equiv 0$, $\delta_1 \equiv 0$, $w_0 = v_0 \equiv 0$.

Step 1. For $j = 1, 2, \dots, m$ do

$$\begin{aligned}\alpha_j &= w_j^T A v_j \\ \hat{v}_{j+1} &= A v_j - \alpha_j v_j - \beta_j v_{j-1} \\ \hat{w}_{j+1} &= A^T w_j - \alpha_j w_j - \delta_j w_{j-1} \\ \delta_{j+1} &= \sqrt{|\hat{w}_{j+1}^T \hat{v}_{j+1}|} \\ \beta_{j+1} &= \hat{w}_{j+1}^T \hat{v}_{j+1} / \delta_{j+1} \\ w_{j+1} &= \hat{w}_{j+1} / \beta_{j+1} \\ v_{j+1} &= \hat{v}_{j+1} / \delta_{j+1}\end{aligned}$$

End.

If the algorithm does not break down before the completion of m steps, then v_1, \dots, v_m ; and w_1, \dots, w_m form a **biorthogonal system**. That is, $w_i^T v_j = 0$ if $i \neq j$ and $w_i^T v_i = 1$. (For a proof, see Saad (1996), pp. 206).

In other words, if

$$\begin{aligned}V_m &= (v_1, v_2, \dots, v_m), \\ W_m &= (w_1, w_2, \dots, w_m),\end{aligned}\tag{15.4.1}$$

then

$$V_m^T W_m = W_m^T V_m = I.\tag{15.4.2}$$

Furthermore, if

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & 0 \\ \delta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_m \\ 0 & & \delta_m & \alpha_m \end{pmatrix}\tag{15.4.3}$$

Then

$$W_m^T A V_m = T_m,\tag{15.4.4}$$

$$A V_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T,\tag{15.4.5}$$

$$A^T W_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T.\tag{15.4.6}$$

Breakdown of the Lanczos Method.

If for any j , $v_j = 0$, then $\text{span}\{v_1, \dots, v_{j-1}\}$ is an invariant subspace for A . Similarly, if $w_j = 0$, then $\text{span}\{w_1, \dots, w_{j-1}\}$ is an invariant subspace for A^T . However, if neither v_j nor w_j is zero, but $w_j^T v_j = 0$, then we have a breakdown (see Wilkinson (1965, pp. 389)). In that case, the look-ahead Lanczos idea has to be applied. We refer the readers to the papers of Parlett, Taylor, and Liu (1985), and Freund, Gutknecht, and Nachtigal (1993), Brezinski et al. (1991), for theory and implementations of the look-ahead Lanczos method. The method is implemented in QMRPACK.

15.4.1 The Block Lanczos Method.

As in the Arnoldi case, extension of the Lanczos method to the block case is also straightforward. Starting with $n \times p$ block vectors P_1 and Q_1 such that $P_1^T Q_1 = I$, the block Lanczos method generates right and left Lanczos block vectors $\{Q_j\}$ and $\{P_i\}$ of dimension $n \times p$, and a block-tridiagonal matrix

$$T_B = \begin{pmatrix} T_1 & L_2 & & 0 \\ M_2 & T_2 & \ddots & \\ \ddots & \ddots & L_m & \\ 0 & & M_m & T_m \end{pmatrix}$$

such that with

$$P_{[m]} = (P_1, P_2, \dots, P_m)$$

and

$$Q_{[m]} = (Q_1, Q_2, \dots, Q_m)$$

we have

$$Q_{[m]}^T A Q_{[m]} = T_B$$

and

$$\begin{aligned} A Q_{[m]} &= Q_{[m]} T_B + Q_{m+1} M_{m+1} E_m^T \\ A^T P_{[m]} &= P_{[m]} T_B^T + P_{m+1} L_{m+1}^T E_m^T. \end{aligned}$$

The block Lanczos vectors satisfy the three-term recurrences:

$$\begin{aligned} Q_{j+1} M_{j+1} &= A Q_j - Q_j T_j - Q_{j-1} L_j \\ P_{j+1} L_{j+1}^T &= A^T P_j - P_j T_j^T - P_{j-1} M_j^T, \end{aligned}$$

for $j = 1, 2, \dots$ and $Q_0 = P_0 = 0$.

For details of the algorithm, see Bai et al. (2000).

The block Lanczos method breaks down if $P_{j+1}^T Q_{j+1}$ is singular.

In such a situation, an *adaptively blocked Lanczos method* can be used to avoid breakdown. The scheme was proposed by Bai, Day and Ye (1999) for eigenvalue problems. For a description of the ABLE, see Bai et al. (2000). ABLE adaptively changes the block size and maintain the full or semi biorthogonality of the block Lanczos vectors.

15.5 The Krylov Subspace Methods in Control

In this section, we describe how the Arnoldi and Lanczos methods can be used as projection methods to solve some large and sparse control problems listed in the Introduction.

We start with a general discussion on the scopes of using these methods in control theory.

15.5.1 Scopes of using the Krylov Subspace Methods in Control

Since the Krylov subspace methods (such as, the Arnoldi and Lanczos methods) are the projection methods onto K_m , it is only natural to use these methods as the projection techniques to solve large-scale control problems, as has been done in numerical linear algebra.

The idea is then as follows: First, the original large control problem is projected onto an m -dimensional Krylov subspace by constructing a basis of the subspace. The projected smaller problem is then solved using a standard well-established technique. Finally, an approximate solution of the original problem is recovered from the solution of the projected problem.

The solution of the projected problem is constructed such that either a **Galerkin property** is satisfied; that is, the residual is orthogonal to the associated Krylov subspace, or the norm of the residual error is minimized.

The examples of projection methods in control include

- (i) Lanczos methods for controllability and observability problems (Boley and Golub (1991), Boley (1994), Boley and Datta (1996)).
- (ii) Arnoldi methods for Lyapunov and Sylvester equations (Saad (1990), Hu and Reichel (1992), Jaimoukha and Kasenally (1994), Simoncini (1996), Robb   and Sadkane (2002), El Guennouni, Jbilou and Riquet (2002)).
- (iii) Arnoldi methods for partial pole placement (Saad (1988), Datta and Saad (1991))
- (iv) Lanczos and Arnoldi methods for model reduction (Bai, Feldman and Freund (1997), Bai and Freund (1999), Boley (1994), Feldman and Freund (1995a, 1995b, 1995c), Freund (1995, 1997, 1999), Gragg and Lindquist (1983), Grimme, Sorensen and Van Dooren (1996), Gallivan, Grimme and Van Dooren (1994), Jaimoukha and Kasenally (1995, 1997), Su and Craig, Jr. (1991), Antoulas, Sorensen and Gugercin (2001), Antoulas (2002), etc.)

Yet, another way to use the Arnoldi method is to exploit the nice property of the Arnoldi equation

$$AV_m - V_m H_m = h_{m+1,m}(0, 0, \dots, 0, v_{m+1}),$$

which has a **striking resemblance** with the single-output Sylvester-observer equation arising in the construction of the observer (see **Chapter 12**):

$$AX - XH = (0, c).$$

This has been done in Datta and Saad (1991) to develop an Arnoldi method to solve the single-output Sylvester-observer equation.

Some of these algorithms will now be described in some details in the subsequent subsections.

15.5.2 The Controllability and the observability problems and the Krylov Subspace Methods

Because of the nature of the Krylov subspace methods, it is only natural to think of solving the controllability and observability problems using these methods.

We shall discuss this in the context of continuous-time problems only. Let $S_c, S_o, S_{\bar{c}}$, and $S_{\bar{o}}$, denote, respectively the controllable, observable, uncontrollable, and unobservable subspaces, then a classical result in this area states that

$$\begin{aligned} S_c &= K_\infty(A, B) \\ S_{\bar{o}} &= K_\infty(A^T, C^T)^\perp, \end{aligned}$$

where S^\perp denotes the orthogonal complement of the set S , and K_∞ denotes the Krylov subspace of infinite dimension.

Thus, starting with A and B , the Arnoldi method can be used to compute an orthonormal basis of S_c . Similarly, starting with A^T and C^T , an orthonormal basis of the unobservable subspace $S_{\bar{o}}$ can be constructed using the Arnoldi method. Similarly, for the spaces S_c and S_o . Indeed, by computing the orthonormal bases of the various subspaces and their intersections, one can compute the so-called **Kalman decomposition, as follows.**

Define

$$\begin{aligned} S_{c\bar{o}} &= S_c \cap S_{\bar{o}}, \quad S_{co} = S_c \cap S_o, \\ S_{\bar{c}\bar{o}} &= S_{\bar{c}} \cap S_{\bar{o}}, \quad S_{\bar{c}o} = S_{\bar{c}} \cap S_o, \\ T &= (T_{\bar{c}o}, T_{\bar{c}\bar{o}}, T_{co}, T_{c\bar{o}}), \end{aligned}$$

where T_{ab} is a basis for the corresponding S_{ab} . Then the Kalman decomposition $(\hat{A}, \hat{B}, \hat{C})$ of the continuous-time system (A, B, C) can be computed as:

$$\begin{aligned} T^{-1}AT = \hat{A} &= \begin{pmatrix} \hat{A}_{\bar{c}o} & 0 & 0 & 0 \\ * & \hat{A}_{\bar{c}\bar{o}} & 0 & 0 \\ * & 0 & \hat{A}_{co} & 0 \\ * & * & * & \hat{A}_{c\bar{o}} \end{pmatrix}, \\ \hat{B} = T^{-1}B &= \begin{pmatrix} 0 \\ 0 \\ \hat{B}_{co} \\ \hat{B}_{c\bar{o}} \end{pmatrix}, \\ \hat{C} = CT &= (\hat{C}_{\bar{c}o}, 0, \hat{C}_{co}, 0). \end{aligned}$$

The matrix T obtained this way is known to have the best condition number (with respect to 2-norm) among all the transforming matrices giving the Kalman decomposition. For details, see Boley (1994).

Boley and Golub (1991), have described a modified version of the Lanczos method and shown how this method can be used to construct the bases of the above various subspaces.

Computing An Orthonormal Basis for the space $S_a \cap S_b$.

Given two orthonormal bases T_1 and T_2 of the two respective subspaces S_1 and S_2 , an orthonormal basis for the intersection space $S_1 \cap S_2$ is given by $\{T_2v_1, \dots, T_2v_r\}$, where v_1 through v_r are the right singular vectors of the matrix $T_1^T T_2$ corresponding to the multiple singular value 1. For details, see Boley (1994).

15.5.3 Arnoldi Method For Rank-one Lyapunov Equation

We have seen before (**Chapter 7**), when A is a stable matrix, the unique solution of the Lyapunov equation

$$AX + XA^T + BB^T = 0$$

is explicitly given by

$$X = \int_0^\infty e^{tA} BB^T e^{tA^T} dt.$$

Arnoldi Method in Case B is a vector b

In the following, we describe an Arnoldi-based method, suitable for large and sparse computations, for the rank-one Lyapunov matrix equation. That is, we consider the Lyapunov equation $AX + XA^T + bb^T = 0$, where b is a vector.

From the explicit expression above, we then have

$$X = \int_0^\infty e^{tA} bb^T e^{tA^T} dt.$$

In order to use this explicit formula computationally, we need an approximation to $e^{At}b$. An Arnoldi method can be developed to compute $e^{At}b$ by observing that

$$e^{At}b \cong \beta V_m e^{H_m t} e_1,$$

where H_m and V_m are obtained by running m steps of the Arnoldi method using $v_1 = \frac{b}{\|b\|_2}$ as the starting vector, and $\beta = \|b\|_2$ and e_1 is the first column of the $m \times m$ identity matrix. In fact, it can be shown [see Saad (1992a)] that if the minimal polynomial of b is of degree equal to m , the approximation is exact for all t .

Substituting the above expression for $e^{At}b$ in the integral expression of X , we have an approximation X_m to X given by:

$$\begin{aligned} X_m &= V_m \left(\int_0^\infty e^{\tau H_m} (\beta e_1) (\beta e_1^T) e^{\tau H_m^T} d\tau \right) V_m^T \\ &= V_m G_m V_m^T, \text{ where } G_m \text{ is the expression within the parenthesis on the right-hand side,} \\ &\quad \text{and } e_1 \text{ is the first column of an } m \times m \text{ identity matrix } I_m. \end{aligned}$$

If H_m is stable, then G_m satisfies Lyapunov equation

$$H_m G_m + G_m H_m^T + \beta^2 e_1 e_1^T = 0.$$

The above discussions lead to the following algorithm. The algorithm was originally proposed by Saad (1990).

Algorithm 15.5.1 An Arnoldi-method for the Rank-one Lyapunov equation

Inputs: A - The $n \times n$ **stable** state-matrix

b - The control vector

m - A positive integer less than n .

Output: A low-rank approximation X_m to the solution X of the Lyapunov equation: $AX + XA^T + bb^T = 0$

Step 1.

Run m steps of the Arnoldi algorithm with

$$v_1 = \frac{b}{\|b\|_2} = \frac{b}{\beta}.$$

Obtain V_m and H_m .

Step 2.

Solve the $m \times m$ Lyapunov matrix equation:

$$H_m G_m + G_m H_m^T + \beta^2 e_1 e_1^T = 0,$$

using **Algorithm 8.6.2**.

Step 3.

Compute X_m , an approximation to X :

$$X_m = V_m G_m V_m^T.$$

Remarks: Saad reports satisfactory numerical results on a problem of order 800 with values of $m = 5, 10, 15$, and 20 (see also **Example 15.5.2** below.)

The Galerkin Property of the Residual, the Residual Error Norm and the Error Bound for the Approximate Solution

1. Define the residual error $Res(X_m) = AX_m + X_mA^T + bb^T$. Then it can be shown (see Saad (1990)) that $Res(X_m)$ satisfies the **Galerkin property**. More specifically, it can be shown [**Exercise 2**] that $V_m^T Res(X_m) V_m = 0$.
2. The **residual error norm** for the projection solution is given by [**Exercise 2**]:

$$\begin{aligned} \|Res(G_m)\|_F &= \|H_m G_m + G_m H_m^T + \beta^2 e_1 e_1^T\|_F \\ &= \sqrt{2} \|h_{m+1,m} e_1^T G_m\|_F. \end{aligned}$$

3. The relative error

$$\frac{\|X - X_m\|_F}{\|X_m\|_F} \leq \frac{\sqrt{2n} \|A\|_F}{\|\eta\| \|b\|_F^2} \|h_{m+1,m} e_1^T G_m\|_F,$$

where η is the logarithm norm of A .

Remarks:

1. Though X_m is of order n , this matrix can be stored efficiently as the product of low-order matrices V_m, G_m , and V_m^T . Also note that $\text{rank}(X_m) = \text{rank}(G_m) \leq m$.
2. The stability of A (that is A having all its eigenvalues with negative real parts) can be guaranteed if $A + A^T < 0$.
3. If G_m is symmetric, so is X_m . The matrix G_m will be symmetric positive definite if H_m is stable.
4. The residual error norm after m steps : $\|Res(G_m)\|_F = \sqrt{2} \|h_{m+1,m}^T e_1^T G_m\|_F$ can be computed rather cheaply without any extra cost, as the quantities G_m and $h_{m+1,m}$ are all available at the end of m steps.
5. Detailed error bounds of the Arnoldi approximation of $e^{At}v$ are given in Saad (1992b) and Hochbruck and Lubich (1985).

Example 15.5.1 Consider solving $AX + XA^T + b = 0$ with

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -6 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & -7 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad m=2.$$

Step 1. $v_1 = \frac{b}{\|b\|_2}$.

$$V_m = \begin{pmatrix} 0.3780 & -0.7627 \\ 0.3780 & -0.2460 \\ 0.3780 & -0.0738 \\ 0.3780 & 0.0984 \\ 0.3780 & 0.2706 \\ 0.3780 & 0.4429 \\ 0.3780 & 0.2706 \end{pmatrix}, \quad H_m = \begin{pmatrix} -3.4286 & -2.1946 \\ -2.1946 & -3.6392 \end{pmatrix}.$$

Step 2. Solution of the projected Lyapunov equation

$$G_m = \begin{pmatrix} 1.3513 & -0.5163 \\ -0.5163 & 0.3113 \end{pmatrix}.$$

Step 3. The approximate solution

$$X_m = \begin{pmatrix} 0.6718 & 0.4483 & 0.3738 & 0.2993 & 0.2248 & 0.1503 & 0.2248 \\ 0.4483 & 0.3079 & 0.2611 & 0.2143 & 0.1675 & 0.1207 & 0.1675 \\ 0.3738 & 0.2611 & 0.2235 & 0.1860 & 0.1484 & 0.1109 & 0.1484 \\ 0.2993 & 0.2143 & 0.1860 & 0.1576 & 0.1293 & 0.1010 & 0.1293 \\ 0.2248 & 0.1675 & 0.1484 & 0.1293 & 0.1102 & 0.0911 & 0.1102 \\ 0.1503 & 0.1207 & 0.1109 & 0.1010 & 0.0911 & 0.0813 & 0.0911 \\ 0.2248 & 0.1675 & 0.1484 & 0.1293 & 0.1102 & 0.0911 & 0.1102 \end{pmatrix}$$

Verification.

$$Res(X_m) = \begin{pmatrix} 0.5556 & -0.0099 & -0.1983 & -0.2378 & -0.1283 & 0.1302 & 0.7657 \\ -0.0099 & -0.2316 & -0.3055 & -0.2858 & -0.1726 & 0.0343 & 0.3890 \\ -0.1983 & -0.3055 & -0.3412 & -0.3019 & -0.1873 & 0.0023 & 0.2634 \\ -0.2378 & -0.2858 & -0.3019 & -0.2612 & -0.1639 & -0.0099 & 0.1761 \\ -0.1283 & -0.1726 & -0.1873 & -0.1639 & -0.1023 & -0.0025 & 0.1269 \\ 0.1302 & 0.0343 & 0.0023 & -0.0099 & -0.0025 & 0.0247 & 0.1159 \\ 0.7657 & 0.3890 & 0.2634 & 0.1761 & 0.1269 & 0.1159 & 0.3560 \end{pmatrix}$$

and

$$\begin{aligned} \|V_m^T Res(X_m) V_m\|_F &= 1.8018 \times 10^{-15} \\ \|Res(X_m)\|_F &= 1.8431. \end{aligned}$$

This shows that the Galerkin condition is satisfied, as expected.

Example 15.5.2 (A Numerical Experiment).

Algorithm 15.5.1 was run with a sparse matrix of order 400 with varying m from $m = 3$ to $m = 63$. The results are displayed below. Using MATLAB notation, we write

$$A = -diag((1 : 400)) + diag(10, 399) + diag(10, -399)$$

which is a sparse stable 400×400 matrix with entries $-1, \dots, -400$ on the diagonal and 10 on the top right and bottom left corners;

$$b = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \end{pmatrix}^T.$$

The Frobenius norm of the residual matrices $Res(X_m) = AX_m + X_mA^T + bb^T$, $m = 3, 6, 9, \dots$, are shown in the Table below and plotted in Figure 15.2 using logarithmic scale.

Table 15.2

m	$\ Res(X_m)\ _F$	m	$\ Res(X_m)\ _F$	m	$\ Res(X_m)\ _F$
3	16945.574289	24	1.657211	45	0.115513
6	41.085860	27	1.203320	48	0.072463
9	14.814631	30	0.863883	51	0.044698
12	7.840236	33	0.606970	54	0.027126
15	4.841644	39	0.277332	57	0.016196
18	3.251037	36	0.415582	60	0.009511
21	2.292672	42	0.180821	63	0.005490

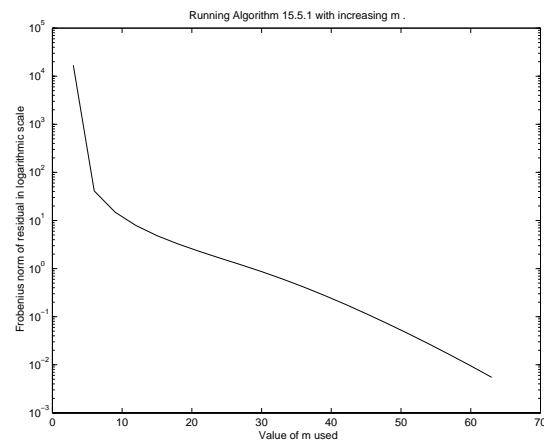


Figure 15.2: Plot of the Frobenius Norm of the Residual Matrices in Logarithmic Scale.

15.5.4 Restarted Arnoldi Method for Lyapunov Equation

If an approximation X_m to the solution of the Lyapunov equation is not satisfactory, the value of m can be increased and the Arnoldi iteration can be repeated (as shown in **Example 15.5.2**). However, as m increases, the cost of computing G_m also increases, and it might become excessive for large values of m . As an alternative to this, it is sometimes suggested that a variable m_1 (a small integer) is introduced and the residual error norm is evaluated at every m_1 iterations of the Arnoldi method. Since the residual norm $\| \text{Res}(G_m) \|_F = \sqrt{2} \| h_{m+1,m} e_m^T G_m \|_F$ can be computed rather cheaply, this can then be used as a stopping criteria.

Algorithm 15.5.2 A Restarted Arnoldi Algorithm for Rank-one Lyapunov Equation

Inputs: A - The $n \times n$ stable state-matrix
 b - The control vector
 m_1 - An integer parameter (usually 3 or 4).
 ϵ - A specified tolerance (> 0).

Output: A low-rank approximation X_m to the solution X of the Lyapunov equation:

$$AX + XA^T + bb^T = 0.$$

Step 0. (Initialization)

Set $l \equiv 0$, and $m \equiv m_1$

Step 1. Normalize the vector b to obtain the starting Arnoldi vector v_1 :

$$v_1 = \frac{b}{\| b \|_2} = \frac{b}{\beta}.$$

Step 2. (Arnoldi method to generate H_m and V_m).

For $k = l + 1, l + 2, \dots, l + m_1$ do

$$\hat{v} = Av_k$$

For $j = 1, 2, \dots, k$ do

$$h_{j,k} = v_j^T \hat{v}$$

$$\hat{v} = \hat{v} - h_{jk} v_j$$

End

$$h_{k+1,k} = \| \hat{v} \|_2$$

$$v_{k+1} = \hat{v} / h_{k+1,k}$$

End.

Step 3. Find the symmetric matrix G_m satisfying the Lyapunov equation of order m :

$$H_m G_m + G_m H_m^T + \beta^2 e_m e_m^T = 0,$$

where $H_m = (h_{ij})$.

Step 4. Compute the residual norm:

$$\| \text{Res}(G_m) \|_F = \sqrt{2} \| h_{m+1,m} e_m^T G_m \|_F$$

Step 5. **Stopping Test:** If $\| \text{Res}(G_m) \|_F > \epsilon$, set $l \equiv l + m_1, m \equiv l + m_1$ and return to Step 2.

Step 6. Form the approximate solution

$$X_m = V_m G_m V_m^T,$$

where $V_m = (v_1, \dots, v_m)$.

Note: The number m in Step 5 gives the count for total number of iterations.

A Perturbation Result (Jaimoukha and Kasenally (1994)).

It can be shown [Exercise 3] that the low-rank approximate solution X_m obtained by Algorithm 15.5.2 satisfies

$$(A - \Delta_m) X_m + X_m (A - \Delta_m)^T + b b^T = 0,$$

where $\Delta_m = v_{m+1} h_{m+1,m} v_{m+1}^T$.

15.5.5 Block Arnoldi Method For Discrete Lyapunov Equation

The Arnoldi algorithms (**Algorithms 15.5.1 and 15.5.2**) described above for the single-input Lyapunov equation

$$AX + XA^T + bb^T = 0$$

can be easily generalized to the multi-input case; that is, when b is a matrix B , using the Block Arnoldi algorithm. We leave this derivation to the reader [Exercise 4]. Instead, we describe below a block Arnoldi algorithm for the multi-input discrete Lyapunov equation:

$$AXA^T - X = -BB^T.$$

The algorithm is analogous to the algorithm for the continuous case and appeared (almost) simultaneously in the work of Boley (1994), and Jaimoukha and Kasenally (1994). In their paper, Jaimoukha and Kasenally (1994) have also described a GMRES-type algorithm for $AX + XB = -C$, by adapting and extending a Hessenberg-Lyapunov solver due to Howland and Senez (1970).

Let A be a discrete-stable matrix. Then, we know [Chapter 7] that the controllability Gramian for the above discrete system

$$C_G = \sum_{i=0}^{\infty} A^i B B^T (A^T)^i$$

satisfies the discrete-Lyapunov equation:

$$AC_G A^T + BB^T = C_G.$$

Let $B = V_1 R$ be the QR factorization of B . Then after m steps of the Block Arnoldi algorithm with V_1 as the starting block of vectors, we generate the block Hessenberg matrix H_m and the orthonormal matrix U_m (**Algorithm 15.3.4**).

Define

$$G_m = \sum_{i=0}^{\infty} (H_m)^i \begin{pmatrix} R \\ 0 \end{pmatrix} (R^T \ 0) (H_m^T)$$

so that G_m satisfies the equation

$$H_m G_m H_m^T + \begin{pmatrix} R \\ 0 \end{pmatrix} (R^T \ 0) = G_m.$$

Then an approximate solution \hat{C}_G to the equation $AC_G A^T + BB^T = C_G$ will be

$$\hat{C}_G = U_m G_m U_m^T.$$

Thus we have the following Arnoldi algorithm for the discrete-time Lyapunov equation:

$$AXA^T - X = -BB^T.$$

Algorithm 15.5.3 A Block Arnoldi Algorithm for Discrete-time Lyapunov Equation

Input: A - The $n \times n$ state-matrix

B - The $n \times p$ control matrix

m - A positive integer ($m < n$).

Output: X_m - A low-rank approximate solution of $AXA^T + BB^T = X$.

Assumption: A is **discrete-stable**; that is, all the eigenvalues of A are inside the unit circle.

Step 1. Find the QR factorization of B :

$$B = V_1 R$$

Step 2. Run m steps of the block Arnoldi algorithm to obtain H_m , U_m , and $H_{m+1,m}$ with V_1 as obtained in Step 1.

Step 3. Obtain an $mp \times mp$ matrix G_m by solving the discrete Lyapunov equations using the Schur-method described in Chapter 8:

$$H_m G_m H_m^T + \begin{pmatrix} R \\ 0 \end{pmatrix} (R^T \ 0) = G_m.$$

Step 4. Compute the approximate solution $X_m = U_m G_m U_m^T$.

The Galerkin Property of the Residual, the Residual Error Norm and the Error Bound for the Approximate Solution

1. As in the continuous-time case, it can be shown [**Exercise 9**] that the residual

$$\text{Res}(X_m) = AX_mA^T - X_m + BB^T$$

satisfies the **Galerkin property**:

$$U_m^T \text{Res}(X_m) U_m = 0.$$

Furthermore, the **residual error norm** for the solution of the projected problem is given by

$$\|\text{Res}(G_m)\|_F = \left\| H_{m+1,m} E_m^T G_m \begin{pmatrix} \sqrt{2}H_m^T & E_m H_{m+1,m}^T \end{pmatrix} \right\|_F$$

For a proof, see Jaimoukha and Kasenally (1994).

- 2 Let $\rho(M)$ denote the spectral radius of M . Then the following **error bound for the approximate solution X_m** holds:

$$\|X - X_m\|_2 \leq 2 \frac{\|B\|_2^2 \theta^2 \delta^{2k}}{1 - \delta^2},$$

for some δ satisfying $\rho(A) < \delta < 1$, and $\rho(H_m) < \delta < 1$ and some constant θ depending on A and the choice of δ .

It can be shown that the error bound converges to zero as m increases. See Boley (1994).

Remarks

Remarks similar to those of the continuous-time rank-one Lyapunov equation also holds for this algorithm. Thus we note:

1. $\text{Rank } (X_m) = \text{Rank } (G_m) \leq mp$.
2. X_m is symmetric if G_m is so.
3. Since the residual error norm at the end of m steps can be obtained rather cheaply, this again can be used as a stopping criterion. Thus, if $\| \text{Res}(G_m) \|_F$ is greater than a prescribed error tolerance ϵ , the Arnoldi process can be restarted (as in Algorithm 15.5.2) either by increasing the dimension m or by introducing a fixed variable m_1 , and evaluating the residual error norm at every m_1 iterations. We leave this as an exercise [**Exercise 5**].

Example 15.5.3 Consider solving $AXA^T + BB^T = X$, with using **Algorithm 15.5.3**

$$A = \begin{pmatrix} -10 & 0 & 0 & 0 & 0 & 1 \\ 0 & -2 & 0 & 0 & 1 & 0 \\ 0 & 0 & -5 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -6 & 0 \\ 0 & 0 & 0 & 0 & 0 & -8 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 3 \\ 3 & 4 \\ 3 & 1 \\ 4 & 4 \\ 5 & 6 \\ 8 & -1 \end{pmatrix}, \quad p = 2, \quad m = 2.$$

Step 1. The QR factorization of B gives

$$V_1 = \begin{pmatrix} -0.1775 & -0.2883 \\ -0.2662 & -0.3629 \\ -0.2662 & 0.0548 \\ -0.3549 & -0.2982 \\ -0.4437 & -0.5120 \\ -0.7099 & 0.6567 \end{pmatrix}, \quad R = \begin{pmatrix} -11.2694 & -5.2354 \\ 0 & -7.1827 \end{pmatrix}$$

Step 2. Block-Arnoldi algorithm yields

$$H_m = \begin{pmatrix} -6.2835 & 1.3308 & -1.1622 & -0.5056 \\ 1.6768 & -6.4910 & -1.1550 & -1.4459 \\ -1.4051 & -1.9261 & -4.6093 & -0.1716 \\ 0 & -1.8687 & -0.4515 & -7.4351 \end{pmatrix}, \quad U_m = \begin{pmatrix} -0.1775 & -0.2883 & -0.3083 & -0.7015 \\ -0.2662 & -0.3629 & 0.6943 & 0.2408 \\ -0.2662 & 0.0548 & 0.3086 & -0.5514 \\ -0.3549 & -0.2982 & 0.2210 & -0.0831 \\ -0.4437 & -0.5120 & -0.5214 & 0.3560 \\ -0.7099 & 0.6567 & -0.0836 & 0.1109 \end{pmatrix}$$

Step 3. Solution of the projected equation

$$G_m = \begin{pmatrix} -7.2569 & -4.6978 & 4.4144 & 0.9512 \\ -4.6978 & -4.1972 & 3.4121 & 0.8870 \\ 4.4144 & 3.4121 & -3.0792 & -0.7194 \\ 0.9512 & 0.8870 & -0.7194 & -0.1896 \end{pmatrix}$$

Step 4. Approximate solutions

$$X_m = \begin{pmatrix} -0.0701 & -0.5321 & -0.2015 & -0.3918 & -0.3233 & -0.1748 \\ -0.5321 & -7.3382 & -2.3915 & -5.0080 & -3.7523 & -1.1318 \\ -0.2015 & -2.3915 & -0.8800 & -1.6780 & -1.2364 & -0.6497 \\ -0.3918 & -5.0080 & -1.6780 & -3.4493 & -2.5989 & -0.9105 \\ -0.3233 & -3.7523 & -1.2364 & -2.5989 & -2.0165 & -0.6467 \\ -0.1748 & -1.1318 & -0.6497 & -0.9105 & -0.6467 & -0.9688 \end{pmatrix}$$

Verification of the Galerkin Condition:

$$U_m^T \text{Res}(X_m) U_m = 10^{-12} \begin{pmatrix} -0.2984 & -0.1261 & -0.0067 & 0.0624 \\ -0.0706 & 0.0151 & -0.0113 & 0.0518 \\ 0.0535 & -0.0704 & 0.0089 & -0.0051 \\ 0.1252 & 0.0022 & -0.0075 & -0.0110 \end{pmatrix}$$

Example 15.5.4 (A Numerical Experiment)

A numerical experiment was performed on **Algorithm 15.3** using random data: $A = \text{rand}(100)$, $B = \text{rand}(100, 20)$. The results of the experiment are given below.

Table 15.3

n	p	m	k (block size)	$\ U_m^T \text{Res}(X_m)^* U_m\ $
6	2	2	2	$3.6185e - 013$
10	2	1	2	$7.8557e - 016$
		2	2	$1.9424e - 014$
100	20	1	20	$7.5551e - 011$
		2	20	$1.7312e - 010$
		4	20	$5.4616e - 011$

15.5.6 Arnoldi Methods for Sylvester Equation

Let A , B , and C be the matrices of order n . (Note that the matrix B here is not the usual control matrix.)

We have seen in **Chapter 8** that the Sylvester equation

$$AX - XB = C$$

can be written as the following linear systems of equations:

$$(I \otimes A - B^T \otimes I)x = c$$

where \otimes denotes the Kronecker product, and x and c are vectors with n^2 components.

More precisely,

$$e_j^T X e_k = e_{j+n(k-1)}^T x, \quad 1 \leq j, k \leq n$$

$$e_j^T C e_k = e_{j+n(k-1)}^T c, \quad 1 \leq j, k \leq n$$

Thus, once the above linear system is solved, the entries of the matrix X can be obtained from those of the vector x using the above formula. We can use either the Arnoldi method of Galerkin type or the GMRES method, described in the previous section, to solve the system. However, when the matrices are large, a formidable amount of computer storage will be required to store a basis of the Krylov subspace $K_m(I \otimes A - B^T \otimes I, r_0)$. Here r_0 is the initial residual vector (see below). Recently Hu and Reichel (1992) have described a Galerkin method to solve

the above system that requires a considerable reduced amount of storage space, by exploiting the structure of the matrix of the system.

We first present the **Hu-Reichel method of Galerkin-type**, and then show how this method may be modified to obtain a minimized residual method, also proposed by these authors.

A Galerkin Method for Solving the Sylvester Equation $AX - XB = C$

The basic idea is to replace the Krylov subspace $K_m(I \otimes A - B^T \otimes I, r_0)$ with a subspace of the form $K_m(B^T, g) \otimes K_m(A, f)$ for certain vectors f and g .

The vectors f and g are chosen so that the initial residual vector $r_0 = b - Ax_0$, where x_0 is the initial approximate solution, lies in the Krylov subspace $K_m(B^T, g) \otimes K_m(A, f)$. Once these vectors are chosen, and the m -steps of the Arnoldi method is run on A and B^T with the Arnoldi vectors $v_1 = f/\|f\|_2$ and $v_1 = g/\|g\|_2$, respectively, generating the matrices H_A, H_B, V_m , and W_m , it can be shown that the columns of the matrix $W_m \otimes V_m$ form an orthonormal basis of $K_m(B^T, g) \otimes K_m(A, f)$. This basis now can be used. The method is outlined below in some details.

Let x_0 be an initial approximation to the solution of the linear system. We wish to determine a correction vector $z_0 \in K_m(B^T, g) \otimes K_m(A, f)$ such that the residual vector

$$r_1 = c - (I \otimes A - B^T \otimes I)x_1$$

associated with the new solution-vector

$$x_1 = x_0 + z_0$$

is orthogonal to $K_m(B^T, g) \otimes K_m(A, f)$.

The correction vector z_0 can be written as

$$z_0 = (W_m \otimes V_m)y_0$$

for some vector $y_0 \in \mathbb{R}^{m^2}$.

Let $\tilde{r}_0 = (W_m \otimes V_m)^T r_0$. Then substituting the above expression of z_0 into $r_1 = r_0 - (I \otimes A - B^T \otimes I)z_0$ and requiring that $(W_m \otimes V_m)^T r_1 = 0$, one obtains the linear system of equations:

$$(I \otimes H_A - H_B \otimes I)y_0 = \tilde{r}_0.$$

Once y_0 is obtained by solving the above linear system; z_0 , x_1 and the corresponding residual vector r_1 , can be computed using their above respective expressions. If $\|r_1\|_2$ is not sufficiently small, then the method is restarted by setting $x_0 \equiv x_1$ and $r_0 \equiv r_1$, with new vectors f , and g .

Determining the vectors f and g

Ideally, f and g should be determined so that

$$r_0 \in K_m(B^T, g) \otimes K_m(A, f).$$

However, such vectors are difficult to determine. Instead, the vectors f and g are sought such that $\|r_0 - g \otimes f\|_2$ is small or equivalently, $\|R_0 - fg^T\|_F$ is small, where the entries of the $n \times n$ matrix R_0 are given by:

$$e_j^T R_0 e_k = e_{j+n(k-1)}^T r_0, \quad 1 \leq j, k \leq n.$$

Now, $\min_{f,g \in \mathbb{R}^n} \|R_0 - fg^T\|_F = \|R_0 - \sigma_1 q_\ell q_r^T\|_F$,

where σ_1 is the largest singular value of R_0 , and q_ℓ and q_r are the associated left and right singular vectors. An **approximate** solution to the above minimization problem can be computed as follows:

Let $f = (f_1, f_2, \dots, f_n)^T$ and $g = (g_1, g_2, \dots, g_n)^T$. Introduce $T(f, g) = \|R_0 - fg^T\|_F^2$. Then

$$\text{For } \forall j, \quad \frac{\partial T}{\partial f_j} = 0 \Rightarrow f = \frac{R_0 g}{\|g\|^2}$$

$$\text{For } \forall j, \quad \frac{\partial T}{\partial g_j} = 0 \Rightarrow g = \frac{R_0^T f}{\|f\|^2}$$

This gives us the following scheme to compute f and g :

If $\|R_0\|_1 \geq \|R_0\|_\infty$, then determine
 $g = \frac{R_0^T f}{\|f\|^2}$, taking f as a column of R_0
of largest norm.
Else determine $f = \frac{R_0 g}{\|g\|^2}$, taking g as
a row of R_0 of largest norm.

Solving $(I \otimes H_A - H_B \otimes I)y_0 = \tilde{r}_0$

In order to solve the above linear system, we can transform the matrices H_A and H_B to real Schur Forms (RSF) or to upper triangular forms using the QR iteration scheme and then solve the transformed triangular system.

Assume for simplicity, that they are transformed into upper triangular matrices. Let

$$H_A = Q_A R_A Q_A^*, \quad H_B = Q_B R_B Q_B^*,$$

where Q_A and Q_B are unitary and R_A and R_B are upper triangular matrices. Then the system $(I \otimes H_A - H_B \otimes I)y_0 = \tilde{r}_0$ is equivalent to the system:

$$(I \otimes R_A - R_B \otimes I)y'_0 = r'_0,$$

where $r'_0 = (Q_B \otimes Q_A)^* \tilde{r}_0$ and $y'_0 = (Q_B \otimes Q_A)^* y_0$. The above is an $m^2 \times m^2$ upper triangular system that can be solved by back substitution.

Remark: Note that the above linear system is equivalent to solving the Sylvester equation:

$$H_A Y - Y H_B^T = \tilde{R}_0,$$

where $\text{vec}(Y) = y_0$ and $\text{vec}(\tilde{R}_0) = \tilde{r}_0$. The above Sylvester equation can now be solved using the Hessenberg-Schur method described in **Chapter 8**.

Computing $r_1 = c - (I \otimes A - B^T \otimes I)x_1$

Computing r_1 using the above expression may be quite expensive unless the matrices A and B are very sparse. Instead, a cheaper way is to evaluate the following expression:

$$r_1 = r_0 - (W_m \otimes V_{m+1} \tilde{H}_A)y_0 + (W_{m+1} \tilde{H}_B \otimes V_m)y_0.$$

If the computation can be properly organized, then the flop-count can be reduced from $4n^3$ to $(2m + 1)n^2$.

Algorithm 15.5.4 A Restarted Arnoldi Algorithm for the Sylvester Equation $AX - XB = C$ (Galerkin type).

Input: x_0 - An initial approximation

ϵ - Tolerance (> 0)

m - Dimension of the Krylov Subspace

Output: An approximate solution X_0 of the Sylvester equation $AX - XB = C$.

Step 1. Compute $r_0 \equiv c - (I \otimes A - B^T \otimes I)x_0$.

Step 2. If $\|r_0\|_2 \leq \epsilon$, then compute the approximate solution matrix X_0 of the equation $AX - XB = C$ from the entries of x_0 .

Step 3. Choose f and g using the scheme described above. Using the Arnoldi algorithm, compute the orthonormal bases of $K_{m+1}(A, f)$ and $K_{m+1}(B^T, g)$; that is, obtain $H_A, H_B, \tilde{H}_A, \tilde{H}_B, V_m, V_{m+1}, W_m, W_{m+1}$.

Step 4. Compute $\tilde{r}_0 = (W_m \otimes V_m)^T r_0$.

Step 5. Determine Q_A and R_A from H_A , and Q_B and R_B from H_B , by Schur factorizations. That is, find $Q_A, R_A; Q_B, R_B$ such that $H_A = Q_A R_A Q_A^*$ and $H_B = Q_B R_B Q_B^*$. Compute $r'_0 = (Q_B \otimes Q_A)^* \tilde{r}_0$.

Step 6. Solve the triangular or the quasi-triangular system:

$$(I \otimes R_A - R_B \otimes I)y'_0 = r'_0$$

Step 7. Compute $y_0 = (Q_B \otimes Q_A)y'_0$.

Step 8. *Compute the correction vector*

$$z_0 = (W_m \otimes V_m)y_0$$

Step 9. *Update the solution*

$$x_0 \equiv x_0 + z_0$$

Step 10. *Compute the updated residual vector:*

$$r_0 \equiv r_0 - (W_m \otimes V_{m+1}\tilde{H}_A)y_0 + (W_{m+1}\tilde{H}_B \otimes V_m)y_0.$$

Step 11. *Compute $\|r_0\|_2$ and go to Step 2.*

Remark: (Breakdown of the Algorithm).

A breakdown of the algorithm occurs when the matrix of the linear system in Step 6 becomes singular. In this case, one can either reduce m or restart the algorithm with a different f and g , e.g., random vectors. The same action should be taken when $\dim K_m(A, f) < m$ or $\dim K_m(B^T, g) < m$. From numerical view point, this means that some subdiagonal entry of H_A is small in the first case, and in the second case, some subdiagonal entry of H_B is small.

A Minimal Projected Residual Algorithm

A minimal projected residual algorithm (in the spirit of the GMRES method) for solving the Sylvester equation can be obtained, just by replacing the Step 6 of the above algorithm by the solution of a least-squares problem.

Thus if

$$\begin{aligned} \tilde{R}_A &= \tilde{Q}_A^* \tilde{H}_A Q_A \\ \tilde{R}_B &= \tilde{Q}_B^* \tilde{H}_B \tilde{Q}_B, \end{aligned}$$

where $\tilde{Q}_A = \text{diag}(Q_A, 1)$ and $\tilde{Q}_B = \text{diag}(Q_B, 1)$, then replace Step 6 by the following:

Solve the least-squares problem:

$$\min_{y'_0 \in R^{m^2}} \|r'_0 - (I_{m+1,m} \otimes \tilde{R}_A - \tilde{R}_B \otimes I_{m+1,m})y'_0\|_2,$$

where $r'_0 = (\tilde{Q}_B \otimes \tilde{Q}_A)^* \tilde{r}_0$, where $\tilde{r}_0 = (W_{m+1} \otimes V_{m+1})^T r_0$.

We invite the readers to fill-in the details [Exercise 7].

15.5.7 Block Arnoldi Methods for Sylvester Equation

Recently block Lanczos and Arnoldi methods have been developed for the Sylvester equation (see Simoncini (1996), El Guennouni, Jbilou and Riquet (2002) and Robb   and Sadkane (2002)). Simoncini extended the Hu-Reichel algorithm to the block form, El Guennouni et al. proposed new Krylov subspace algorithms based on block Arnoldi and Lanczos methods, and Robb   and Sadkane gave detailed convergence analysis of the GMRES and Galerkin-type block Arnoldi methods for the Sylvester equation, with a particular attention to breakdown and stagnation in these methods. In particular, they showed that the breakdown in the block Arnoldi method has a positive consequence method that the approximate solutions become exact solutions.

We state now two block Arnoldi methods for the Sylvester equation $AX - XB = C$, taken from Robb   and Sadkane (2002); one is Galerkin-type and the other the GMRES-type. The matrices $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{p \times p}$, where $p \ll n$. Both these algorithms seek an approximate solution $X_m = X_0 + U_m Z_m$, where X_0 is an initial approximation and Z_m is obtained by solving a small Sylvester equation in case of the Galerkin-type method, and a matrix minimization problem in case of the GMRES-type method.

Let $S(X) = AX - XB$ and the residual $R_0 = C - S(X_0)$. Assume that $\text{rank}(R_0) = q \leq p$. Consider the QR factorization of $R_0 = V_1 \Lambda_1$ where $V_1 \in \mathbb{R}^{n \times q}$, $V_1^T V_1 = I_q$, and $\Lambda_1 \in \mathbb{R}^{q \times p}$ upper triangular.

Suppose that m steps of the block Arnoldi method has been performed with V_1 as the starting vector, generating the matrices U_m , H_m , and \tilde{H}_m .

The Galerkin-type method finds X_m^G such that $X_m^G = X_0 + U_m Z_m$, where $Z_m \in \mathbb{R}^{mq \times p}$ solves the Sylvester equation:

$$H_m Z_m - Z_m B = \Lambda, \quad \Lambda = \begin{pmatrix} \Lambda_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{mq \times p}$$

The GMRES-type method finds the solution X_m^{GM} such that

$$X_m^{GM} = X_0 + U_m Z_m,$$

where $Z_m \in \mathbb{R}^{mq \times p}$ solves the matrix least squares problem:

$$\min_{Z \in \mathbb{R}^{mq \times p}} \|\bar{\Lambda} - \bar{S}_m(Z)\|_F, \text{ where}$$

$$\bar{S}_m(Z) = \tilde{H}_m(Z) - \begin{pmatrix} Z \\ 0 \end{pmatrix} B, \quad \bar{\Lambda} = \begin{pmatrix} \Lambda_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{(m+1)q \times p}$$

and

$$\tilde{H}_m = \begin{pmatrix} H_m \\ 0, \dots, 0 & H_{m+1,m} \end{pmatrix} \in \mathbb{R}^{(m+1)q \times mp}.$$

The above discussion leads to the following Algorithm

Algorithm 15.5.5 Block Arnoldi Methods for the Sylvester Equation $AX + XB = C$

Inputs: A - An $n \times n$ matrix
 B - An $p \times p$ matrix, $p << n$
 C - An $n \times p$ matrix
 X_0 - An initial approximation

Outputs: X_m - An approximate solution of $AX - XB = C$.

Assumption: The spectra of A and B are disjoint.

Step 1. Compute the residual matrix $R_0 = C - (AX_0 - X_0B)$.

Step 2. Obtain an upper triangular matrix Λ_1 by computing the full rank QR factorization of R_0 :

$$R_0 = V_1 \Lambda_1.$$

Step 3. Run m steps of the Arnoldi algorithm with V_1 as obtained in Step 2 and obtain U_m, H_m , and \tilde{H}_m .

Step 4. (*Galerkin-type*): Compute the approximate solution

$$X_m^G = X_0 + U_m Z_m,$$

obtaining Z_m by solving the Sylvester equation:

$$H_m Z_m - Z_m B = \Lambda,$$

where

$$\Lambda = \begin{pmatrix} \Lambda_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{mq \times p}.$$

(*GMRES-type*): Compute the approximate solution

$$X_m^{GM} = X_0 + U_m Z_m,$$

obtaining Z_m by solving the minimization problem:

$$\min_{Z \in \mathbb{R}^{mq \times p}} \|\bar{\Lambda} - \bar{S}_m(Z)\|_F,$$

where

$$\bar{\Lambda} = \begin{pmatrix} \Lambda_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{(m+1)q \times p}$$

and

$$\bar{S}_m(Z) = \tilde{H}_m Z - \begin{pmatrix} Z \\ 0 \end{pmatrix} B.$$

Notes:

- (i) The small Sylvester equation in Step 4 for the Galerkin-type method can be solved by using the **Hessenberg-Schur method** described in Chapter 8.
- (ii) The minimization problem in Step 4 for the GMRES-type method can be solved by computing the real Schur factorization of the matrix B , and then solving a triangular matrix least-square problem.

Details are left as an exercise [**Exercise 16**].

- (iii) (**Residuals and Restart**). It can be shown (Robb   and Sadkane (2002)) that the residuals $R_m^{GM} = S(X_m^{GM}) - C$ and $R_m^G = S(X_m^G) - C$ satisfy:

$$\|R_m^{GM}\|_F = \|\bar{\Lambda} - \bar{S}_m(Z_m)\|_F$$

and

$$\|R_m^G\|_F = \|H_{m+1,m}Z_m^L\|_F,$$

where Z_m^L is the last $q \times p$ block of Z_m .

Using these easily computed residuals, Algorithm 15.5.5 should be periodically restarted with $X_0 = X_m^G$ or $X_0 = X_m^{GM}$, where $X_m^G | X_m^{GM}$ is the last computed approximate solution with Galerkin/GMRES method.

Convergence Analysis. A detailed Convergence analysis of the above algorithms have been given by Robb   and Sadkane (2002). We state the main points of their analysis.

1. The GMRES algorithm converges if the field of values of A and B are disjoint.
2. If the Galerkin algorithm converges, then the GMRES algorithm also converges. However, if GMRES stagnates (i.e. $\|R(X_m^{GM})\|_F = \|R_0\|_F$), then the Galerkin algorithm fails.

Note: It is assumed that R_0 and the parameter m are the same in both these algorithms.

3. (**Breakdown**). If the block Arnoldi algorithm breaks down at iteration m ; that is, if $H_{m+1,m} = 0$, then the approximate solutions computed by GMRES and the Galerkin algorithm are exact; that is, $X_m^G = X_m^{GM} = X$.

Galerkin Condition: The Galerkin Condition is satisfied with X_m^G : $U_m^T R_m^G = 0$.

Example 15.5.5 Consider solving $AX - XB = C$ using Algorithm 15.5.5 with $m = 2$ and

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -6 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & -7 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 3 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad X_0 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Step 1.

$$R_0 = \begin{pmatrix} 2 & 1 & 3 \\ 4 & 5 & 6 \\ 5 & 5 & 8 \\ 6 & 6 & 8 \\ 7 & 7 & 9 \\ 8 & 8 & 10 \\ 7 & 7 & 9 \end{pmatrix}$$

Step 2. V_1 , obtained by rank revealing QR factorization of R_0 :

$$V_1 = \begin{pmatrix} -0.1438 & 0.7032 & 0.3294 \\ -0.2877 & -0.2743 & -0.6824 \\ -0.3836 & 0.5680 & -0.4942 \\ -0.3836 & 0.0077 & 0.0471 \\ -0.4315 & -0.1314 & 0.1882 \\ -0.4795 & -0.2705 & 0.3294 \\ -0.4315 & -0.1314 & 0.1882 \end{pmatrix}$$

Step 3. Block Arnoldi with V_1 as the starting block-vectors:

$$H_m = \begin{pmatrix} -4.5816 & -1.4186 & 0.7424 & -0.1997 & 1.1540 & 0 \\ -1.4186 & -2.6283 & 1.2440 & -1.4643 & 0 & 0 \\ 0.7424 & 1.2440 & -2.6096 & -0.2692 & -0.2597 & 0.5256 \\ -0.1997 & -1.4643 & -0.2692 & -5.2178 & 1.9362 & 0.9902 \\ 1.1540 & 0 & -0.2597 & 1.9362 & -2.8284 & -0.2909 \\ 0 & 0 & 0.5256 & 0.9902 & -0.2909 & -5.4401 \end{pmatrix}$$

$$U_m = \begin{pmatrix} -0.1438 & 0.7032 & 0.3294 & -0.1833 & -0.5735 & -0.1132 \\ -0.2877 & -0.2743 & -0.6824 & -0.1833 & -0.5735 & -0.1132 \\ -0.3836 & 0.5680 & -0.4942 & 0.0959 & 0.5070 & -0.1358 \\ -0.3836 & 0.0077 & 0.0471 & 0.4188 & -0.1416 & 0.5435 \\ -0.4315 & -0.1314 & 0.1882 & 0.3652 & -0.0629 & 0.2202 \\ -0.4795 & -0.2705 & 0.3294 & 0.1216 & 0.0660 & -0.7128 \\ -0.4315 & -0.1314 & 0.1882 & -0.7746 & 0.2384 & 0.3225 \end{pmatrix}$$

Step 4. Solution by the Galerkin-type method:

$$\Lambda = \begin{pmatrix} -15.5346 & -15.6784 & -20.8567 \\ -0.8075 & -1.7850 & 0 \\ 1.0118 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad Z_m = \begin{pmatrix} 2.7401 & 2.6131 & 2.5418 \\ -1.0094 & -0.4027 & -0.4067 \\ -0.2461 & 0.2647 & 0.2745 \\ 0.4666 & 0.1852 & 0.0232 \\ 0.9925 & 0.6836 & 0.3281 \\ 0.0056 & 0.0166 & 0.0078 \end{pmatrix}$$

$$X_m^G = \begin{pmatrix} -0.8404 & 0.0003 & 0.2456 \\ 0.0012 & -0.2497 & -0.0003 \\ 0.0445 & 0.0002 & -0.1741 \\ -0.0125 & -0.0032 & 0.0023 \\ 0.0130 & 0.0034 & -0.0022 \\ -0.0036 & -0.0010 & 0.0006 \\ -0.2192 & 0 & 0.0710 \end{pmatrix}$$

Verification: $\|R_m^G\|_F = \|AX_m^G - X_m^G B - C\|_F = 0.1053$ **(Residual Norm)**

$$U_m^T R_m^G = 10^{-14} \begin{pmatrix} -0.5249 & -0.4573 & 0.3641 \\ 0.0972 & 0.3039 & -0.1008 \\ -0.0115 & 0.0372 & -0.0865 \\ -0.1739 & 0.0688 & -0.0022 \\ -0.3369 & -0.3279 & -0.0043 \\ -0.0111 & -0.0111 & 0.0051 \end{pmatrix} \quad \text{(Galerkin Condition)}$$

Restart with $X_0 = X_m$.

Step 1.

$$R_0 = 10^{-1} \begin{pmatrix} 0.0318 & 0.0100 & -0.0019 \\ 0.0318 & 0.0100 & -0.0019 \\ 0.0382 & 0.0120 & -0.0023 \\ -0.6016 & -0.1895 & 0.0365 \\ 0.7607 & 0.2396 & -0.0462 \\ -0.2483 & -0.0782 & 0.0151 \\ -0.0159 & -0.0050 & 0.0010 \end{pmatrix}$$

Step 2. $\text{rank}(R_0) = 1$ and V_1 obtained by rank-revealing QR factorization of R_0 :

$$V_1 = \begin{pmatrix} -0.0317 & -0.0317 & -0.0381 & 0.5998 & -0.7585 & 0.2475 & 0.0159 \end{pmatrix}^T$$

Step 3. Block Arnoldi with V_1 as the starting block-vectors:

$$H_m = \begin{pmatrix} -4.6941 & 0.6004 \\ 0.6004 & -4.6046 \end{pmatrix}$$

$$U_m = \begin{pmatrix} -0.0317 & -0.1424 \\ -0.0317 & -0.1424 \\ -0.0381 & -0.1075 \\ 0.5998 & 0.6934 \\ -0.7585 & 0.3865 \\ 0.2475 & -0.5384 \\ 0.0159 & -0.1667 \end{pmatrix}$$

Step 4. Solution by the Galerkin-type Method:

$$\Lambda = \begin{pmatrix} -0.1003 & -0.0316 & 0.0061 \\ 0 & 0 & 0 \end{pmatrix}, \quad Z_m = 10^{-1} \begin{pmatrix} 0.1840 & 0.0476 & -0.0322 \\ 0.0206 & 0.0043 & -0.0053 \end{pmatrix}$$

$$X_m^G = \begin{pmatrix} -0.8412 & 0.0001 & 0.2457 \\ 0.0003 & -0.2500 & -0.0001 \\ 0.0436 & 0 & -0.1739 \\ 0 & 0 & 0 \\ -0.0001 & 0 & 0 \\ -0.0002 & 0 & 0.0001 \\ -0.2192 & 0 & 0.0711 \end{pmatrix}$$

Verification: $\|R_m^G\|_F = \|AX_m^G - X_m^G B - C\|_F = 0.0025$ (Residual Norm)

$$U_m^T R_m^G = 10^{-15} \begin{pmatrix} -0.0059 & 0.0036 & 0.0070 \\ -0.0086 & 0.8049 & 0.0037 \end{pmatrix} \quad (\text{Galerkin Condition})$$

The following Table below shows how the residual norm decreases with i successive restartings:

Table 15.3

i	$\ AX_m - X_m B - C\ _F$	i	$\ AX_m - X_m B - C\ _F$
1	0.10533325043831	4	0.00026546894274
2	0.00247924225545	5	0.00014261918287
3	0.00109241878919	6	0.00003783546561

15.5.8 Arnoldi-Method for Sylvester-Observer Equation (single-output case)

We have just seen how the Arnoldi method can be used as a projection technique to solve the Sylvester matrix equation. It turns out that the Arnoldi method can also be used to solve a

single-output Sylvester observer equation

$$A^T X - XH = (0, c),$$

where c is a vector, by exploiting the nice property of the Arnoldi equation (**Equation 15.3.3**).

In this section, we describe a method due to Datta and Saad (1991) for doing this.

Recall from Chapter 12 that the Sylvester-observer equation, arising in the construction of Luenberger observer, is:

$$TA - FT = GC,$$

where A , and C are given such that (A, C) is observable; and the matrices T , F , and G are to be found satisfying the constraints:

- (i) (F, G) is controllable.
- (ii) T has full rank.
- (iii) F has an arbitrary spectrum.

Here we consider the “transpose” of this equation:

$$A^T T^T - T^T F^T = C^T G^T$$

or

$$A^T X - XH = C^T G^T$$

In the single-output case, the equation reduces to $A^T X - XH = c^T g^T$, where g^T is a row vector (**Note that the output vector c is a row-vector**).

Suppose H is constructed such that it is an unreduced upper Hessenberg matrix, then the simplest choice of g^T , is $g^T = (0, 0, \dots, 0, 1)$, because, then (H, g^T) is observable. This choice of d^T reduces the equation above to $A^T X - XH = (0, 0, \dots, 0, c^T)$.

The method to be described solves the following problem:

Given an $n \times n$ matrix A , an n -vector c^T , a set of self-conjugate numbers μ_1, \dots, μ_m ($m \leq n$), find an $n \times m$ **orthonormal** matrix X_m , and an $m \times m$ Hessenberg matrix H_m having the set $\{\mu_1, \dots, \mu_m\}$ as its spectrum such that

$$A^T X_m - X_m H_m = (0, c^T).$$

In order to solve this Sylvester-observer equation, we start by observing the striking resemblance between the equation to be solved and the Arnoldi equation (**Equation 15.3.3**):

$$AV_m - V_m H_m = h_{m+1,m}(0, 0, \dots, 0, v_{m+1}).$$

However, in the equation to be solved, the vector c^T is arbitrarily given, and in the Arnoldi equation, the last vector on the right hand side, $h_{m+1,m}v_{m+1}$ is constructed by the Arnoldi algorithm.

Since the Arnoldi method is uniquely determined by the starting Arnoldi vector v_1 , a question, naturally arises:

Can the Arnoldi vector v_1 be chosen such that the corresponding last vector v_{m+1} in the Arnoldi method is equal to c^T ?

To answer to the above question, we need the following Theorem. For a proof of the Theorem, see Saad (1992).

Theorem 15.5.1 *If the m steps of Arnoldi method is run on an $n \times n$ matrix A^T , then apart from a multiplicative scalar, the polynomial p_m such that $v_{m+1} = p_m(A^T)v_1$ is the characteristic polynomial of the Hessenberg matrix H_m . Moreover, this polynomial minimizes the norm $\|q(A^T)v_1\|$ over all monic polynomials $q(x)$ of degree m .*

Since the characteristic polynomial of H_m must be

$$q(t) = (t - \mu_1)(t - \mu_2) \cdots (t - \mu_m),$$

we see by the above Theorem (Theorem 15.5.1) that the vector $q(A^T)v_1$ must be proportional to the vector c^T . The desired vector v_1 is therefore such that

$$v_1 = \alpha[q(A^T)]^{-1}c^T,$$

where α is a normalizing scalar that makes the 2-norm of v_1 equal to unity. This alone will not, of course, guarantee that the constructed Hessenberg matrix will have the desired spectrum. However, application of the above Theorem again immediately suggests the following procedure:
Run $m - 1$ steps of Arnoldi's method with the above choice of v_1 , get the orthonormal matrix V_m and the first $m - 1$ columns of the Hessenberg matrix, and then construct the last column in such a way that the resulting Hessenberg matrix has the eigenvalues μ_1, \dots, μ_m . This will complete the algorithm.

The last column of the Hessenberg matrix can be computed using a variation of a single-input eigenvalue assignment algorithm described in Chapter 11.

The proposed algorithm for computing the solution of the Sylvester-observer equation can now be written as follows:

Algorithm 15.5.6 An Arnoldi Algorithm for Single-output Sylvester-observer Equation

Inputs: A - The $n \times n$ state-matrix.
 m - An integer less than or equal to m
 $\{\mu_1, \dots, \mu_m\}$ - A set of m complex numbers, closed under complex conjugation.
 c - The output vector (A row vector)

Outputs: H_m - An $m \times m$ Hessenberg matrix having $\{\mu_1, \dots, \mu_m\}$ as the spectrum and X_m - An $n \times m$ orthonormal matrix satisfying:

$$A^T X_m - X_m H_m = (0, c^T).$$

Step 1. Solve the linear system

$$q(A^T)x = c^T,$$

and compute $v_1 = x/\|x\|$; where $q(t) = (t - \mu_1)(t - \mu_2)\dots(t - \mu_m)$.

Step 2. Run $m - 1$ steps of the Arnoldi method on A^T with v_1 as the initial vector to generate V_m and the first $m - 1$ columns of H_m . Let \tilde{H}_{m-1} denote the matrix of the first $m - 1$ columns of H_m .

Step 3. Find a column vector y such that

$$\Omega([\tilde{H}_{m-1}, y]) = \Omega(H_m) = \{\mu_1, \dots, \mu_m\},$$

where $\Omega(K)$ denotes the spectrum of the matrix K .

Step 4. Compute $\alpha = (c')^T c' / \|c'\|^2$, where c' is the last column of $A^T V_m - V_m H_m$.

Step 5. Set $X_m = (1/\alpha)V_m$.

Note: Observe that the solution obtained by this algorithm has the nice additional property of being orthonormal.

Solving the Equation $q(A^T)x = c^T$

The obvious bottleneck in the algorithm is the numerical solution of the polynomial system in step 1. An effective method for solution of this system is now considered. The obvious way of solving $q(A^T)x = c$ is to solve successively the linear systems:

$$(A^T - \mu_i I)x_{i+1} = x_i, \quad i = 0, 1, \dots, m-1,$$

with $x_0 = c^T$; the solution x is the final solution x_m . This is not a satisfactory method for at least two reasons. First, it is too expensive. Direct methods are excluded in view of the

fact that we assume that A is very large. For small m , preconditioned iterative methods are feasible, but a new preconditioner must be found for each of the m linear systems. The second reason is the potential instability of the iteration. For a discussion on preconditioned iterative methods, see Saad (1996), and Axelsson (1994).

A Partial Fraction Approach

We will now show an effective way of solving the above linear system by reformulating the problem slightly and resorting, once again, to the GMRES procedure. We seek to compute the vector $x = f(A^T)c^T$, where f is the rational function

$$f(t) = \frac{1}{q(t)} = \frac{1}{\prod_{i=1}^m (t - \mu_i)}.$$

An important observation is the $f(t)$ can be rewritten as

$$f(t) = \sum_{i=1}^m \frac{1}{q'(\mu_i)(t - \mu_i)},$$

where $q'(\mu_j)$ is the derivative of q at μ_j and is equal to

$$q'(\mu_j) = \prod_{i=1, \dots, m, i \neq j} (\mu_i - \mu_j).$$

Hence the desired solution x can be written as

$$x = \sum_{i=1}^m \frac{1}{q'(\mu_i)} (A^T - \mu_i I)^{-1} c^T.$$

In other words, all we need is to solve the m independent linear systems:

$$(A^T - \mu_i I)x_i = c^T, i = 1, \dots, m,$$

and then obtain the solution x by the linear combination of x_1, \dots, x_m as follows:

$$x = \sum_{i=1}^m \frac{1}{q'(\mu_i)} x_i.$$

Solving $(A^T - \mu_i)x_i = c^T, i = 1, \dots, m$

In the following discussion we assume that k steps of the Arnoldi process have been performed on A^T , starting with $v_1 = c^T / \|c^T\|$. At the end of the k steps we obtain:

- (1) an orthonormal basis $V_k \equiv [v_1, v_2, \dots, v_k]$ of the Krylov subspace $\text{span } \{c, Ac, \dots, A^{k-1}c\}$, together with

- (2) the representative equation in that subspace

$$A^T V_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T.$$

In order to solve several systems of the form $(A^T - \mu_i I)x_i = c^T$, we make the important observation that the above relation can also be written for any shift μ_i , since

$$(A^T - \mu_i I)V_k = V_k(H_k - \mu_i I) + h_{k+1,k}v_{k+1}e_k^T.$$

Hence the same approximation can be used. Note that we solve k independent small $m \times m$ linear systems with the matrices $H_k - \mu_i I$. The bulk of the work is in constructing V_k , and this is done only once.

Choosing the Eigenvalues and the Numerical Stability of the Partial Fraction Approach

A detailed stability (numerical) property of the partial fraction approach for solving the system

$$q(A^T)x = c^T$$

has been studied in Calvetti, Gallopoulos, and Reichel (1995), Calvetti and Reichel (1997) and Calvetti, Lewis and Reichel (2001). Their crucial observation is that the performance of the scheme depends heavily upon how the m numbers μ_1, \dots, μ_m are chosen. Certain choices may lead to a poor accuracy. They have suggested the choice of these numbers as the equidistant points on a circle or on the zeros of a certain Chebyshev polynomial.

More specifically, let $\tau_o = \min_t Re(t)$ and $\rho = \max_t Im(t)$, where t runs over the entire spectrum of A and assume that $\tau_o \leq 0$ and $\rho > 0$. Then their suggestion is to choose μ_j to be the zeros of the Chebyshev polynomial of the first kind of degree m for the interval $[\tau + j\rho, \tau - j\rho]$ for some $\tau < \tau_o$. The details and the reasoning of this choice can be found in Calvetti, Lewis and Reichel (2001).

Replacing the Last Column of a Hessenberg Matrix to Achieve a Desired Spectrum (Implementation of Step 3.)

In this section, we present a very simply procedure for replacing the last column of an $m \times m$ unreduced upper Hessenberg matrix so that the resulting matrix has a desired set of eigenvalues $\{\mu_1, \mu_2, \dots, \mu_m\}$. The method is a variation of the recursive single-input pole-assignment algorithm proposed by Datta (1987) (**Algorithm 11.2.1**).

Let H_m be an unreduced $m \times m$ lower Hessenberg matrix. Construct a set of normalized vectors $\{l_k\}$ as:

$$l_1 = e_1 = (1, 0, 0, \dots, 0)^T.$$

For $i = 1, 2, \dots, m-1$ do

$$\hat{l}_{i+1} = (H_m^T - \mu_i I)l_i,$$

$$l_{i+1} = \frac{\hat{l}_{i+1}}{\|\hat{l}_{i+1}\|},$$

End

$$s = (H_m^T - \mu_m I)l_m.$$

Set

$$f = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \alpha \end{pmatrix}, \text{ where } \alpha = \prod_{i=1}^{m-1} h_{i,i+1}.$$

Then

Theorem 15.5.2 *The upper Hessenberg matrix $H_m^T - sf^T$ has the spectrum $\{\mu_1, \dots, \mu_m\}$.*

Proof. Similar to that of **Algorithm 11.2.1**.

Remarks:

Methods for the multi-output problem, analogous to the one just presented, have not been developed yet.

However, a full Arnoldi-type of method ($m = n$) for the construction of an orthogonal solution to the multi-output Sylvester-observer equation has recently been developed by Datta and Hetti (1997).

15.5.9 Arnoldi-Method for Continuous-time Algebraic Riccati Equation

It is certainly plausible that Arnoldi methods that we have just now described for the Lyapunov and Sylvester equations can also be developed for the algebraic Riccati equations.

Following Jaimoukha and Kasenally (1994), we describe below an idea for the continuous-time algebraic Riccati equation, which certainly deserves merit for further development in the discrete-time case.

Consider the continuous-time algebraic Riccati equation (CARE):

$$XA + A^T X - XBB^T X + LL^T = 0$$

described in Chapter 13.

Suppose that m -steps of the block Arnoldi method (**Algorithm 15.3.4**) have been taken constructing the matrices $U_m, V_{m+1}, H_m, H_{m+1,m}$, and L_m starting with the block Arnoldi vector V_1 , generated from the QR factorization of L .

Suppose G_m satisfies the $mp \times mp$ projected equation:

$$G_m H_m + H_m^T G_m - G_m B_m B_m^T G_m + L_m L_m^T = 0,$$

where

$$U_m^T B = B_m, \text{ and } U_m L_m = L.$$

Then it has been shown by Jaimoukha and Kasenally (1994) that, provided that the associated Hamilton matrix has no purely imaginary eigenvalues, the residual norm satisfies the Galerkin condition; that is

$$U_m^T \text{Res}(X_m) U_m = 0,$$

where $X_m = U_m G_m U_m^T$, and

$$\text{Res}(X_m) = X_m A + A^T X_m - X_m B B^T X_m + L L^T$$

This observation suggests the following algorithm.

Algorithm 15.5.7 An Arnoldi Algorithm for Continuous-time Algebraic Riccati Equation (Galerkin-type).

Inputs: The system matrices A and B , and an $n \times r$ matrix L .

Output: An approximate solution X_m of the CARE: $X A + A^T X - X B B^T X + L L^T = 0$.

Step 1. Compute $U_m, H_m, H_{m+1,m}$ by running m steps of the block Arnoldi method starting with V_1 given by: $L = V_1 R$ (**QR factorization of L**).

Step 2. Define B_m by $U_m^T B = B_m$ and L_m by $U_m L_m = L$.

Step 3. Solve the projected equation for G_m :

$$G_m H_m + H_m^T G_m - G_m B_m B_m^T G_m + L_m L_m^T = 0$$

Step 4. Compute the low-rank approximation X_m of X :

$$X_m = U_m G_m U_m^T$$

Remarks:

- As described earlier, a re-started Arnoldi method can be developed [Exercise 10] if the solution after m steps of the Arnoldi iterations is not satisfactory, by noting that the residual error norm after m steps can be computed cheaply as:

$$\|\text{Res}(G_m)\|_F = \sqrt{2} \|H_{m+1,m} E_m^T G_m\|_F.$$

- A similar method may possibly be developed for the discrete-time algebraic Riccati equation:

$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$

We leave this as a **research problem**.

Example 15.5.6 Consider finding a low-rank approximation of the solution of $XA + A^T X - XBB^T X + LL^T = 0$, with $m = 2, p = 2$, and

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 7 \end{pmatrix}$$

$$B = \begin{pmatrix} -0.4326 & -0.0376 & -0.1364 \\ -1.6656 & 0.3273 & 0.1139 \\ 0.1253 & 0.1746 & 1.0668 \\ 0.2877 & -0.1867 & 0.0593 \\ -1.1465 & 0.7258 & -0.0956 \\ 1.1909 & -0.5883 & -0.8323 \\ 1.1892 & 2.1832 & 0.2944 \end{pmatrix}, \quad L = \begin{pmatrix} -1.3362 & -1.4410 \\ 0.7143 & 0.5711 \\ 1.6236 & -0.3999 \\ -0.6918 & 0.6900 \\ 0.8580 & 0.8156 \\ 1.2540 & 0.7119 \\ -1.5937 & 1.2902 \end{pmatrix}$$

Step 1. The block Arnoldi algorithm with V_1 obtained from the QR factorization of L : $L = V_1 R$ gives

$$V_1 = \begin{pmatrix} -0.4172 & 0.5571 \\ 0.2230 & -0.2153 \\ 0.5069 & 0.2145 \\ -0.2160 & -0.3067 \\ 0.2679 & -0.3123 \\ 0.3915 & -0.2574 \\ -0.4976 & -0.5826 \end{pmatrix}, \quad R = \begin{pmatrix} 3.2029 & 0.2320 \\ 0 & -2.4128 \end{pmatrix}$$

$$U_m = \begin{pmatrix} -0.4172 & 0.5571 & -0.6386 & 0.1509 \\ 0.2230 & -0.2153 & 0.0233 & -0.1763 \\ 0.5069 & 0.2145 & -0.3198 & -0.6172 \\ -0.2160 & -0.3067 & 0.0529 & 0.2857 \\ 0.2679 & -0.3123 & -0.3417 & 0.2411 \\ 0.3915 & -0.2574 & -0.4842 & 0.4323 \\ -0.4976 & -0.5826 & -0.3678 & -0.4884 \end{pmatrix}$$

$$H_m = \begin{pmatrix} 4.6577 & 1.2348 & -0.0979 & 1.8412 \\ 1.2348 & 3.5293 & 2.3120 & -0.0000 \\ -0.0979 & 2.3120 & 4.1344 & 0.3943 \\ 1.8412 & 0 & 0.3943 & 4.4887 \end{pmatrix}.$$

Step 2.

$$B_m = \begin{pmatrix} -0.6222 & -0.9047 & 0.1123 \\ -0.5852 & -1.3438 & 0.1827 \\ -0.4096 & -0.8003 & 0.0791 \\ -0.1091 & -1.3702 & -1.2088 \end{pmatrix}$$

$$L_m = \begin{pmatrix} 3.2029 & 0.2320 \\ 0 & -2.4128 \\ 0 & 0 \\ 0.0000 & 0.0000 \end{pmatrix}.$$

Step 3: Solution of the projected Riccati equation:

$$G_m = \begin{pmatrix} 165.8395 & -25.5827 & -199.1183 & 31.0136 \\ -25.5827 & 15.6132 & 19.1699 & -8.5244 \\ -199.1183 & 19.1699 & 271.6279 & -36.0167 \\ 31.0136 & -8.5244 & -36.0167 & 10.1367 \end{pmatrix}.$$

Step 4. The approximate solution

$$X_m = \begin{pmatrix} 38.4805 & 6.3498 & 48.8044 & -14.7975 & 41.8943 & 61.6413 & 0.4928 \\ 6.3498 & 6.8406 & 22.5802 & -4.9596 & 24.6299 & 34.5244 & -1.7706 \\ 48.8044 & 22.5802 & 99.9787 & -26.4442 & 96.0492 & 137.3621 & -5.1115 \\ -14.7975 & -4.9596 & -26.4442 & 7.9093 & -22.5540 & -32.7188 & 1.5029 \\ 41.8943 & 24.6299 & 96.0492 & -22.5540 & 101.7769 & 144.3091 & -5.2122 \\ 61.6413 & 34.5244 & 137.3621 & -32.7188 & 144.3091 & 204.9250 & -7.5577 \\ 0.4928 & -1.7706 & -5.1115 & 1.5029 & -5.2122 & -7.5577 & 3.3063 \end{pmatrix}.$$

Verification:

$$Res(X_m) =$$

$$\begin{pmatrix} -41.7445 & -54.5758 & -52.3990 & -36.6734 & -71.3969 & -17.9356 & 18.4533 \\ -54.5758 & -38.4516 & -91.7767 & 3.2399 & -100.5122 & -106.8096 & 10.1210 \\ -52.3990 & -91.7767 & -47.3717 & -80.9724 & -85.0597 & 40.7114 & 32.9713 \\ -36.6734 & 3.2399 & -80.9724 & 45.3681 & -69.8429 & -140.6445 & -5.6159 \\ -71.3969 & -100.5122 & -85.0597 & -69.8429 & -127.0498 & -16.7070 & 34.6013 \\ -17.9356 & -106.8096 & 40.7114 & -140.6445 & -16.7070 & 211.4409 & 43.1806 \\ 18.4533 & 10.1210 & 32.9713 & -5.6159 & 34.6013 & 43.1806 & -2.1913 \end{pmatrix}$$

and

$$\left| \|Res(X_m)\|_F - \sqrt{2} \|H_{m+1,m} E_m^T G_m\|_F \right| = 1.3074 \times 10^{-12} ;$$

$$\|U_m^T Res(X_m) U_m\|_F = 9.6413 \times 10^{-11} .$$

Thus the residual matrix satisfies the Galerkin condition, as expected.

15.5.10 Arnoldi Method for Partial Eigenvalue Assignment

As mentioned in Chapter 1, the pole assignment (or eigenvalue assignment) problem arises in stabilization of control systems. Mathematically this means that if the eigenvalues of the original system matrix A do not have some desired characteristics, then a matrix K is constructed such that the eigenvalues of the resulting system matrix $A - BK$ can be relocated at desirable locations. In practical situations, however, some eigenvalues of A are “good”, and what is then actually needed, is changing only the “bad” eigenvalues of the system, while keeping all the good eigenvalues in their places or at least not making them unstable. This leads to the **Partial Eigenvalue Assignment Problem** in control theory.

Partial Eigenvalue Assignment Problem for Multi-input Time Invariant Linear Systems

Let A be an $n \times n$ large and sparse matrix with the spectrum $\{\lambda_1, \dots, \lambda_k, \lambda_{k+1}, \dots, \lambda_n\}$, $k < n$. Let $B \in \mathbb{R}^{n \times m}$. Then the partial pole assignment problem is the problem of finding a feedback matrix $K \in \mathbb{R}^{m \times n}$ such that the spectrum of the modified matrix $M = A - BK$ is

$$\Omega(M) = \{\mu_1, \mu_2, \dots, \mu_k; \lambda_{k+1}, \dots, \lambda_n\},$$

where $S = \{\mu_1, \mu_2, \dots, \mu_k\}$ is a prescribed set of k eigenvalues to be assigned and both $S = \{\mu_1, \mu_2, \dots, \mu_k\}$ and $S_1 = \{\lambda_1, \lambda_2, \dots, \lambda_k\}$ are assumed to be closed under complex conjugation. Moreover, if there is a multiple eigenvalue in the set S_1 , then it is assumed that it appears in the set as many times as its multiplicity.

We describe below a projection method due to Saad (1988). The method is based on computing an orthonormal basis for the left invariant subspace associated with the k eigenvalues that are to be reassigned.

The required orthonormal basis of the left invariant subspace associated with the k eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_k$ may be obtained by computing the **partial Schur factorization for A^T** :

$$A^T Q = QR$$

where Q is an $n \times k$ orthonormal matrix and R is a $k \times k$ quasi-triangular matrix. The columns of the matrix Q form the required basis; they are called the **Schur-vectors**. Details can be found in Saad (1988), Saad (1992).

Saad's Projection Method

Let $A^T Q = QR$ be the partial Schur decomposition of A^T associated with the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_k$. We look for a solution of the form

$$K = (QG)^T$$

where $G \in \mathbb{R}^{k \times m}$ is to be computed.

Now consider

$$\begin{aligned}
M^T Q &= [A^T - K^T B^T] Q \\
&= QR - QGB^T Q \\
&= Q[R - GS_0^T]
\end{aligned}$$

where $S_0^T = B^T Q$.

Write

$$\begin{aligned}
M^T Q &= Q[R^T - S_0 G^T]^T \\
&= QC_k^T
\end{aligned}$$

where $C_k = R^T - S_0 G^T$.

The equation $M^T Q = QC_k^T$ means that the choice of $K^T = QG$ makes the subspace spanned by Q also invariant under M^T . The k eigenvalues of M^T (and hence of M) associated with this invariant subspace are the eigenvalues of the matrix C_k . If we then can find a matrix G such that the matrix $R^T - S_0 G^T$ has the given spectrum $\{\mu_1, \mu_2, \dots, \mu_k\}$, then the matrix M also will have $\mu_1, \mu_2, \dots, \mu_k$ as its eigenvalues.

For the complete solution of the partial eigenvalue assignment problem, we still have to guarantee that the other eigenvalues of M are the same as those of the corresponding eigenvalues of A . The following theorem guarantees that.

Theorem 15.5.3 (Partial Pole-Placement Theorem)

Let G be the feedback matrix such that the spectrum of $R^T - S_0 G^T$ is $\{\mu_1, \dots, \mu_k\}$, and let K be given by $K = (QG)^T$. Then $M = A - BK$ has the eigenvalues $\{\mu_1, \mu_2, \dots, \mu_k, \lambda_{k+1}, \lambda_{k+2}, \dots, \lambda_n\}$.

Proof: Let $W = [w_1, w_2, \dots, w_{n-k}]$ where w_1, w_2, \dots, w_{n-k} are the remaining Schur vectors of A^T associated with the remaining eigenvalues $\lambda_{k+1}, \lambda_{k+2}, \dots, \lambda_n$.

From $M^T Q = QC_k^T$, we have, $Q^T M^T Q = C_k^T$ and $W^T M^T Q = W^T QC_k^T = 0$. Therefore,

$$\begin{aligned}
Q^T M^T W &= Q^T (A^T - QGB^T) W \\
&= Q^T A^T W + E^T,
\end{aligned}$$

where $E = -W^T BG^T$.

Therefore, if we denote $X = [Q, W]$ we have

$$X^T M X = \begin{pmatrix} C_k & 0 \\ W^T A Q + E & W^T A W \end{pmatrix},$$

showing that the eigenvalues of M are $\{\mu_1, \mu_2, \dots, \mu_k; \lambda_{k+1}, \dots, \lambda_n\}$.

Algorithm 15.5.8 A Projection Algorithm for Partial Pole-Placement

- Inputs:**
- A - The $n \times n$ state-matrix
 - B - The $n \times m$ control matrix ($m \leq n$).
 - S - The set $\{\mu_1, \dots, \mu_k\}$ of k eigenvalues to be assigned
- Outputs:**
- K - An $m \times n$ feedback matrix such that the spectrum of $A - BK$ is $\Omega(A - BK) = \{\mu_1, \dots, \mu_k, \lambda_{k+1}, \dots, \lambda_n\}$ where $\lambda_{k+1}, \dots, \lambda_n$ are the $(n - k)$ eigenvalues of A that are required to remain unchanged.

Step 1. Compute the partial Schur decomposition

$$A^T Q = QR$$

associated with the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_k$.

Step 2. Compute $S_0 = Q^T B$ and solve the projected $k \times k$ eigenvalue assignment problem. That is, find a matrix G such that

$$\Omega(R^T - S_0 G^T) = \{\mu_1, \mu_2, \dots, \mu_k\},$$

using a standard multi-input eigenvalue assignment method presented in Chapter 11.

Step 3. Form the feedback matrix

$$K = (QG)^T$$

Example 15.5.7 Consider the partial pole placement problem for the pair (A, B) given by

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & -2 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -5 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & -6 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & -7 \end{pmatrix}, \quad B = \begin{pmatrix} 0.5444 & 0.8373 \\ 0.3976 & 1.6924 \\ 0.0305 & 1.0503 \\ 1.4936 & 0.4053 \\ 0.8902 & 1.3443 \\ 1.8636 & 1.6762 \\ 0.9320 & 0.0393 \end{pmatrix}.$$

$$\mathcal{S} = \Omega(A) = \left\{ -10.7082 \quad -10.3246 \quad -5.0000 \quad -4.0000 \quad -3.0000 \quad 2.3246 \quad 2.7082 \right\}.$$

The objective is to replace the two positive eigenvalues 2.3246 and 2.7082 with positive real parts by -1 and -2 , keeping the remaining 5 negative eigenvalues unchanged.

Step 1. The transforming matrix and the partial Schur decomposition, associated with the two positive eigenvalues:

$$Q = \begin{pmatrix} 0 & 0.8507 \\ -0.8112 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.5847 & 0 \\ 0 & 0.5257 \end{pmatrix}, \quad R = \begin{pmatrix} 2.3246 & 0 \\ 0 & 2.7082 \end{pmatrix}.$$

Step 2. The eigenvalue assignment problem for the pair (R^T, S_0) was solved by using MATLAB command **place**:

$$S_0 = \begin{pmatrix} -1.4122 & -2.3530 \\ 0.9531 & 0.7329 \end{pmatrix}, \quad G = \begin{pmatrix} 2.0177 & -2.6239 \\ 9.1741 & -5.5060 \end{pmatrix}.$$

Step 3. The feedback matrix

$$K = \begin{pmatrix} 7.8039 & -1.6369 & 0 & 0 & 0 & -1.1798 & 4.8231 \\ -4.6837 & 2.1286 & 0 & 0 & 0 & 1.5342 & -2.8947 \end{pmatrix}.$$

Verification:

$$\Omega(A - BK) = \left\{ -3.0000 \quad -4.0000 \quad -5.0000 \quad -10.3246 \quad -10.7082 \quad -1.0000 \quad -2.0000 \right\},$$

as we wanted.

Implementation using the Arnoldi method

It is clear from the development of the above algorithm that the partial Schur factorization required in Step 1 is not indispensable. All that is required is an orthonormal basis of the invariant subspace associated with the eigenvalues $\lambda_1, \dots, \lambda_k$. *The Arnoldi method can certainly be used for that purpose.* Indeed, the k steps of the Arnoldi method will yield an orthonormal matrix V_k whose vectors form an orthonormal basis of the Krylov subspace $K_k(A, v_1) = \text{span}\{v_1, Av_1, \dots, A^{k-1}v_1\}$ and the $k \times k$ Hessenberg matrix H_k such that

$$V_k^T A V_k = H_k.$$

This Hessenberg matrix H_k can then be used in place of the matrix R in the above algorithm, if the k eigenvalues to be replaced correspond to those of H_k . We leave the details of implementation of Algorithm 15.5.8 using the Arnoldi method as an Exercise [Exercise 12].

Remarks:

1. In the recent Ph.D. dissertation by Sarkissian (2001), new theoretical and computational results for both partial eigenvalue and eigenstructure assignments have been given. The theoretical results include conditions for existence and uniqueness of solutions of these

problems. Besides these theoretical results, computational algorithms have been developed to solve those problems whose major computational kernels are solutions of small linear algebraic systems and Sylvester matrix equations. Thus, these algorithms, although not Krylov subspace-based, are practical for large problems.

For details, see Sarkissian (2001) and the MTNS' 2002 paper by Datta and Sarkissian (2002).

2. An Implicitly Restarted Arnoldi (IRA) method for partial eigenvalue assignment problem has been proposed in Calvetti, Lewis, and Reichel (2001).

15.5.11 Lanczos and Arnoldi Methods for Model Reduction

In Chapter 14 we have described model reduction procedures via **internal balancing** and the **Schur decomposition of A** .

These procedures require constructions of the controllability and observability Grammians. Recall that for a stable continuous-time system the controllability Grammian C_G and the observability Grammian O_G satisfy respectively, the Lyapunov equations:

$$\begin{aligned} AC_G + C_G A^T + BB^T &= 0 \\ A^T O_G + O_G A + C^T C &= 0. \end{aligned}$$

Similarly, for a discrete-time system, the corresponding Lyapunov equations are:

$$\begin{aligned} AC_G A^T + BB^T &= C_G \\ A^T O_G A + C^T C &= O_G. \end{aligned}$$

The internal balancing of the system can be achieved by solving these large Lyapunov equations using the Arnoldi methods for these equations, described earlier.

However, since the Schur-method for model reduction requires the transformation of the matrix $C_G O_G$ to real Schur form, the method will not be practical for large and sparse systems.

An alternative approach, using Markov parameters, pursued in the literature for model reduction, might be more appropriate for large and sparse systems.

Suppose that, starting with the $n \times n$ system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx, \end{aligned}$$

a system of order k has been computed with the matrices H_k , B_k , and C_k such that the first few Markov parameters of the original model match with those of the k th order model; then the k th order model can be taken as the reduced-order model.

Intuitively, it is clear that a Krylov subspace method will be a perfect candidate for constructing such a reduced order model.

Indeed, several Krylov subspace methods for model reduction have been developed in recent years. These include the Padé via Lanczos (PVL) approach (Van Dooren (1995), Feldman

and Freund (1995), Freund (1999), etc.), the interpolation approach (Grimme (1994), Gallivan, Grimme and Van Dooren (1994)) based on the rational Krylov method of Ruhe (Ruhe (1984)), implicitly restarted Lanczos method (Grimme, Sorensen and Van Dooren (1996)) and Arnoldi and implicitly restarted dual Arnoldi methods (Jaimoukha and Kasenally (1995, 1997), Sorensen and Antoulas (2001)).

The PVL technique has been proven to be effective in circuit simulation and the multipoint rational interpolation approach has been successful in moment matching of the transfer function at selected frequencies. The machinery needed to describe these techniques has not been developed here and, therefore, we have to skip the descriptions of these techniques.

We will describe here only Lanczos (scalar version) and Arnoldi methods (block version) for model reduction. The implicit Arnoldi method described here has the important feature that it provides a computable expression for the residual error norm associated with the approximate solution.

Our discussions will be confined mostly to the single-input single-output (SISO) case only. The MIMO (Multi-input, Multi-output) case will be just briefly touched upon. For details, we refer the readers to the survey paper of Freund (1999), (and references therein), and the recent article of Freund in the book by Bai et al. (2000) (Section 7.10.4), which describes the use of the band Lanczos algorithm to the MIMO reduced-order modeling.

We start with the application of the scalar Lanczos algorithm for model reduction.

Lanczos Method for Model Reduction (Single-input, Single-output case).

Consider the SISO (single-input, single-output) linear system:

$$\dot{x} = Ax + bu$$

$$y = cx.$$

Suppose that k steps of the Lanczos algorithm (**Algorithm 15.4.1**) are applied to A with starting vectors parallel to b and c^T obtaining the matrices \hat{A}_k , \hat{b}_k and \hat{c}_k given by

$$\begin{aligned}\hat{A}_k &= W_k^T A V_k \\ \hat{b}_k &= W_k^T b \\ \hat{c}_k &= c V_k\end{aligned}$$

Then it can be shown (**Theorem 15.5.2**) that the reduced-order system defined by $(\hat{A}_k, \hat{b}_k, \hat{c}_k)$ preserves the first $2k$ Markov parameters of the original system. (See Gragg (1974), Gragg and Lindquist (1983)).

Algorithm 15.5.9 A Lanczos Algorithm for Model Reduction

- Inputs:**
- A - The $n \times n$ state-matrix
 - b - The control vector (column vector)
 - c - The output vector (row-vector)
 - k - The dimension of the reduced order model

- Outputs:** A_k, b_k , and c_k , the matrices of the reduced-order model
- Step 0.** Scale the vectors b and c to obtain the vectors v_1 and w_1 such that $w_1^T v_1 = 1$.
- Step 1.** Run k steps of the Lanczos algorithm (**Algorithm 15.4.1**) to generate the matrices W_k and V_k .
- Compute:** $A_k = W_k^T A V_k$, $b_k = W_k^T b$, $c_k = c V_k$.
- Step 2.** Form the reduced-order model (A_k, b_k, c_k) .

Theorem 15.5.4 The matrix A_k and the vectors b_k and c_k generated by the above algorithm are such that the first $2k$ Markov parameters of the original and the reduced-order models are the same, that is,

$$c A^{i-1} b = c_k A_k^{i-1} b_k, i = 1, 2, \dots, 2k.$$

Numerical Disadvantages and Possible Cures

There are several numerical difficulties with the above algorithm.

First, there can be serious “breakdowns” in the Lanczos process due to the ill-conditioning of the submatrices in the system’s Hankel matrix.

Second, though the transient response of the reduced-order model will be very close to that of the original model due to the matching of Markov parameters corresponding to the high frequencies, the steady state error can still be large.

Third, the stability of the reduced-order model is not guaranteed even though the original model is stable.

A Modified Lanczos Scheme for Stability

In a recent paper Grimme, Sorensen, and Van Dooren (1996) have suggested an **implicit restated Lanczos scheme** to stabilize the reduced-order model obtained by the Lanczos method. Their idea is as follows:

Suppose that the reduced-order matrix A_k obtained after k steps of the Lanczos model reduction algorithm (**Algorithm 15.5.9**) is not stable and assume that there are q unstable modes: μ_1, \dots, μ_q . Then the idea is to restart the Lanczos model reduction method with the new starting vectors

$$\bar{v}_1 = \bar{p}_\theta (A - \mu_q I) \cdots (A - \mu_1 I) v_1$$

and

$$\bar{w}_1 = \bar{p}_w (A^T - \mu_q I) \cdots (A^T - \mu_1 I) w_1$$

where \bar{p}_θ and \bar{p}_w are certain scalars.

In this case, q additional standard Lanczos steps (**Algorithm 15.4.1**) are required to obtain the order- k Lanczos factorizations:

$$A \bar{V}_k = \bar{V}_k \bar{T}_k + \bar{r}_k e_k^T$$

and

$$A^T \bar{W}_k = \bar{W}_k (\bar{T}_k)^T + q_k e_k^T,$$

which correspond to the relations (15.4.4)-(15.4.6) with the starting vectors \bar{v}_1 and \bar{w}_1 .

The above Lanczos factorizations are computed implicitly from (15.4.4)-(15.4.6) using a technique analogous to the one in Sorensen (1992).

The **stable** reduced-order model in this case is given by:

$$\hat{A}_k = \bar{T}_k = \bar{W}_k^T A \bar{V}_k, \quad \hat{b}_k = \bar{W}_k^T b$$

and

$$\hat{c}_k = c \bar{V}_k$$

The relations between the modified moments of the original system and the above restarted Lanczos model are:

(Modified Moments): For $i \leq 2k$

$$c\Psi(A)A^{i-1}\Psi(A)b = c\Psi(A)\bar{V}_k(\hat{A}_k)^{i-1}\bar{W}_k^T\Psi(A)b,$$

where

$$\Psi(A) = (A - \mu_1 I)(A - \mu_2 I) \dots (A - \mu_q I).$$

For details and derivations of these results see Grimme, Sorensen and Van Dooren (1996).

The MIMO Case

Consider now the model reduction problem for the MIMO system:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

where, as usual, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{r \times n}$.

Block Lanczos Method for Model Reduction

In case $m = r$, the **block Lanczos** method can be used. Specifically, the following result (see Boley (1994)) can be proved.

Theorem 15.5.5 *Let j steps of the block Lanczos method be applied to the MIMO system (A, B, C) , starting with block vectors generated from the QR decompositions of B and C , obtaining the matrices $P_{[j]}$ and $Q_{[j]}$. Define $\hat{A} = Q_{[j]}^T A P_{[j]}$, $\hat{B} = Q_{[j]}^T B$, $\hat{C} = C P_{[j]}$. Then the reduced-order model $(\hat{A}, \hat{B}, \hat{C})$ has the following properties:*

$$\hat{C}\hat{A}^i\hat{B} = CA^iB \text{ for } i = 0, 1, \dots, 2(j-1).$$

Proof: Proof is left as an exercise [Exercise 15].

Band Lanczos Method for Model Reduction

The band Lanczos method is an extension of the standard nonsymmetric Lanczos method for single vectors to blocks of starting vectors of different sizes. This method is thus ideal for the MIMO Case with $m \neq r$. The detailed description of the algorithm is outside the scope of the book. For description of the algorithm, we refer the readers to the paper by Aliga, Boley, Freund and Hernandez (2000). For application of the band Lanczos algorithm to the MIMO model reduction, see Freund (1999). See also the article of Freund in the book by Bai et al. (2000, pp. 205-216).

It can be shown that the reduced order model is characterized as a certain matrix-Padé approximation of the original model. Thus certain moments of the reduced-order model are matched with those of the original model. For details, see Freund (1999).

Arnoldi and Implicitly Restarted Arnoldi Methods for Model Reduction

As stated in Section 15.5.2, the m -dimensional Krylov subspaces $K_m(A, b) = \text{span}\{b, Ab, \dots, A^{m-1}b\}$ and $L_m(A^T, c^T) = \text{span}\{c^T, A^T c^T, \dots, (A^{m-1})^T c^T\}$ are parts of the controllability and observability subspaces, respectively; and these subspaces can be constructed using the Arnoldi method.

It is, therefore, natural to think of using the Arnoldi method for model reduction.

In this section, following Jaimoukha and Kasenally (1995, 1997), we describe how to do so, via both ordinary and implicitly restarted Arnoldi methods.

First, we recall that the m steps of the Arnoldi algorithm (**Algorithm 15.3.1**), starting with $v_1 = \frac{b}{\|b\|_2}$, produce V_m and H_m such that

$$\begin{aligned} AV_m &= V_m H_m + v_{m+1} h_{m+1,m} e_m^T \\ V_m e_1 &= v_1 = \frac{b}{\|b\|_2} \end{aligned}$$

Writing $v_{m+1} = \tilde{V}_m$, $\tilde{H}_m = h_{m+1,m} e_m^T$ and $l_m = \|b\|_2 e_1$, the above equations become

$$\begin{aligned} AV_m &= V_m H_m + \tilde{V}_m \tilde{H}_m \\ b &= V_m l_m \end{aligned}$$

Since these equations are associated with the $K_m(A, b)$, they are called **controllability-Arnoldi equations**. Analogously, the following equations associated with $L_m(A, c)$:

$$\begin{aligned} A^T W_m &= W_m G_m^T + \tilde{W}_m \tilde{G}_m^T \\ c^T &= W_m k_m^T, \end{aligned}$$

where W_m and G_m , \tilde{W}_m and \tilde{G}_m are produced by m -steps of the Arnoldi method, starting with the starting vector $w_1 = \frac{c^T}{\|c\|_2}$, are called the **observability-Arnoldi equations**.

Note that $(\tilde{G}_m)^T = g_{m,m+1}e_m^T$, $\tilde{W}_m = w_{m+1}$, and $k_m = \|c\|_2 e_1^T$.

Let $T_m = W_m^T V_m$. Assuming that T_m is nonsingular, define the matrices

$$\hat{H}_m = T_m^{-1} W_m^T A V_m, \quad \hat{G}_m = W_m^T A V_m T_m^{-1}.$$

Then the reduced order models

$$(i) (\hat{H}_m, T_m^{-1} W_m^T b, c V_m)$$

and

$$(ii) (\hat{G}_m, W_m^T b, c V_m T_m^{-1})$$

satisfy Galerkin conditions, as shown in the following theorem (**Theorem 15.5.6**).

Notes.

$$(i) T_m^{-1} W_m^T b = (W_m^T V_m)^{-1} W_m^T V_m l_m = l_m$$

$$(ii) c V_m = k_m W_m^T V_m = k_m T_m$$

$$(iii) W_m^T b = W_m^T V_m l_m = T_m l_m$$

$$(iv) c V_m T_m^{-1} = k_m T_m T_m^{-1} = k_m$$

Thus, the reduced-order models (i) and (ii) can be, respectively, rewritten as $(\hat{H}_m, l_m, k_m T_m)$ and $(\hat{G}_m, T_m l_m, k_m)$.

Also note that $\hat{H}_m = H_m + T_m^{-1} W_m^T \tilde{V}_m \tilde{H}_m$ and $\hat{G}_m = G_m + \tilde{G}_m \tilde{W}_m^T V_m T_m^{-1}$.

Thus, \hat{H}_m and \hat{G}_m can be obtained directly from the Arnoldi method.

Theorem 15.5.6 (Galerkin Conditions and Residual Errors)

Let $h_m(s) = (sI - \hat{H}_m)^{-1} l_m$ and $g_m(s) = k_m (sI - \hat{G}_m)^{-1}$.

Then the Galerkin conditions $W_m^T ((sI - A)V_m h_m(s) - b) = 0$, and $(g_m(s) W_m^T (sI - A) - c) V_m = 0, \forall s$ are satisfied. Furthermore, the following results on the residual errors hold

$$r_b = \left\| b - (sI - A)V_m h_m(s) \right\|_\infty = \left\| \begin{bmatrix} T_m^{-1} W_m^T \tilde{V}_m \\ 1 \end{bmatrix} \tilde{H}_m h_m(s) \right\|_\infty$$

and

$$r_c = \left\| c - g_m(s) W_m^T (sI - A) \right\|_\infty = \left\| g_m(s) \tilde{G}_m [\tilde{W}_m^T V_m T_m^{-1} \ 1] \right\|_\infty.$$

Algorithm 15.5.10 An Arnoldi Algorithm for Model Reduction

Inputs: A - The $n \times n$ state-matrix

b, c - The control and output vectors

m - An integer less than n .

Outputs: Reduced-order Model $(\hat{H}_m, l_m, k_m T_m)$ or $(\hat{G}_m, T_m l_m, k_m)$.

Step 1. Perform m steps of the Arnoldi method with (A, b) to obtain $H_m, \tilde{H}_m, V_m, \tilde{V}_m$ and l_m .

Step 2. Perform m steps of the Arnoldi method with (A^T, c^T) to produce $G_m, \tilde{G}_m, W_m, \tilde{W}_m$ and k_m .

Step 3. Form $T_m = W_m^T V_m$
 $\hat{H}_m = T_m^{-1} W_m^T A V_m = H_m + T_m^{-1} W_m^T \tilde{V}_m \tilde{H}_m$
and $\hat{G}_m = W_m^T A V_m T_m^{-1} = G_m + \tilde{G}_m \tilde{W}_m^T V_m T_m^{-1}$

Step 4: Form the reduced-order model $(\hat{H}_m, l_m, k_m T_m)$ or $(\hat{G}_m, T_m l_m, k_m)$. (Note that $l_m = e_1 \|b\|_2$ and $k_m = \|c\|_2 e_1^T$).

Example 15.5.8 Consider the model-reduction using Arnoldi method for $m = 4$ and the model (A, b, c) given by

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & -2 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -5 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & -6 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & -7 \end{pmatrix}$$

$$b = \begin{pmatrix} 0.5444 & 0.3976 & 0.0305 & 1.4936 & 0.8902 & 1.8636 & 0.9320 \end{pmatrix}^T$$

$$c = \begin{pmatrix} 0.8373 & 1.6924 & 1.0503 & 0.4053 & 1.3443 & 1.6762 & 0.0393 \end{pmatrix}$$

Step 1: The Arnoldi method applied to A with $v_1 = \frac{b}{\|b\|_2}$ gives

$$V_m = \begin{pmatrix} 0.1947 & -0.4894 & -0.1058 & -0.6099 \\ 0.1422 & -0.8383 & 0.0454 & 0.3347 \\ 0.0109 & -0.0006 & 0.0136 & 0.0348 \\ 0.5341 & 0.0801 & 0.6736 & 0.3346 \\ 0.3183 & 0.1116 & 0.3738 & -0.6220 \\ 0.6664 & 0.1962 & -0.5300 & 0.1247 \\ 0.3333 & 0.0162 & -0.3349 & 0.0209 \end{pmatrix}$$

$$H_m = \begin{pmatrix} -3.2530 & -4.9819 & 0.0000 & 0.0000 \\ -4.9819 & -4.0353 & 3.9542 & 0.0000 \\ 0 & 3.9542 & -4.8634 & 0.7918 \\ 0 & 0 & 0.7918 & -2.7304 \end{pmatrix}$$

$$l_m = \begin{pmatrix} 2.7965 & 0 & -0.0000 & -0.0000 \end{pmatrix}^T.$$

Step 2: The Arnoldi method applied to A^T with $v_1 = \frac{c^T}{\|c^T\|_2}$ gives

$$W_m = \begin{pmatrix} 0.2724 & -0.0391 & -0.4332 & -0.1563 \\ 0.5505 & 0.6271 & 0.1671 & 0.0225 \\ 0.3416 & -0.2635 & 0.4442 & 0.6983 \\ 0.1318 & -0.1380 & 0.1403 & 0.0485 \\ 0.4373 & -0.5779 & 0.2902 & -0.5745 \\ 0.5452 & 0.0383 & -0.5084 & 0.0759 \\ 0.0128 & 0.4258 & 0.4741 & -0.3862 \end{pmatrix}$$

$$G_m = \begin{pmatrix} -0.1972 & 3.6340 & 0 & 0 \\ 3.6340 & -3.9315 & -3.3479 & 0 \\ 0 & -3.3479 & -7.9434 & 1.8808 \\ 0.0000 & 0.0000 & 1.8808 & -3.4823 \end{pmatrix}$$

$$k_m = \begin{pmatrix} 3.0743 & 0 & 0 & 0 \end{pmatrix}$$

. **Step 3:** The matrices T_m , \hat{H}_m and \hat{G}_m are

$$T_m = \begin{pmatrix} 0.7122 & -0.4284 & -0.0402 & -0.1296 \\ -0.0116 & -0.5676 & -0.4428 & 0.5516 \\ -0.0692 & 0.0232 & 0.3731 & 0.1485 \\ -0.2547 & 0.0057 & -0.0659 & 0.5021 \end{pmatrix}$$

$$\hat{H}_m = \begin{pmatrix} -3.2530 & -4.9819 & 0.0000 & 11.5009 \\ -4.9819 & -4.0353 & 3.9542 & 13.7328 \\ 0 & 3.9542 & -4.8634 & -5.8791 \\ 0 & 0 & 0.7918 & 3.9668 \end{pmatrix}$$

$$\hat{G}_m = \begin{pmatrix} -0.1972 & 3.6340 & 0 & 0 \\ 3.6340 & -3.9315 & -3.3479 & -0.0000 \\ 0 & -3.3479 & -7.9434 & 1.8808 \\ 2.3360 & -3.5113 & -1.2444 & 3.8874 \end{pmatrix}$$

Step 4. Reduced-order models $(\hat{H}_m, l_m, k_m T_m)$ and $(\hat{G}_m, T_m l_m, k_m)$, where

$$k_m T_m = \begin{pmatrix} 2.1896 & -1.3171 & -0.1235 & -0.3984 \end{pmatrix}$$

$$T_m l_m = \begin{pmatrix} 1.9918 & -0.0323 & -0.1936 & -0.7123 \end{pmatrix}^T$$

Verification: It can be shown that, for $s = 1$, for instance,

$$W_m^T ((sI - A) V_m h_m(s) - b) = 10^{-14} \begin{pmatrix} -0.0604 & -0.1221 & -0.1887 & -0.2887 \end{pmatrix}^T$$

$$(g_m(s) W_m^T (sI - A) - c) V_m = 10^{-14} \begin{pmatrix} -0.0111 & 0.2123 & 0.1110 & -0.0708 \end{pmatrix}.$$

Thus, the Galerkin conditions are verified.

The table below shows that the Markov parameters of both reduced systems agree with the ones of the original system.

Table 15.4 (Matching of Markov Parameters.)

i	$cA^{i-1}b$	$k_m T_m \hat{H}_m^{i-1} l_m$	$k_m \hat{G}_m^{i-1} T_m l_m$
1	6.123204	6.123204	6.123204
2	-1.568586	-1.568586	-1.568586
3	89.833148	89.833148	89.833148
4	-401.858936	-401.858936	-401.858936
5	3712.889240	3712.889240	3712.889240
6	-305.959145×10^2	-305.959145×10^2	-305.959145×10^2
7	285.048034×10^3	285.048034×10^3	285.048034×10^3

Remarks: At the end of Step 4, one can test the residual error norms given in Theorem 15.5.6. If either of these norms is less than a given error tolerance, say ϵ , one can stop. If not, m can be increased and the Arnoldi process be continued with the increased value of m .

Unfortunately, the reduced-order models obtained by Algorithm 15.5.10 may be unstable even if the original model is stable. Furthermore such models might contain redundant or undesirable modes.

Jaimoukha and Kasenally (1997) have described a restarted Arnoldi framework which may be employed to remove these difficulties.

It is to be noted, however, that the implicitly restarted Arnoldi algorithm proposed by Jaimoukha and Kasenally is different than the ordinary restarted algorithms available in the literature (and described earlier in this chapter in Section 15.5.4).

The present implicit scheme differs from the ones described earlier in the sense that the controllability and observability Arnoldi equations are preserved under this scheme, which in turn, can be used to obtain computable error expressions similar to the ones in Theorem 15.5.6. Furthermore, the implicitly restarted reduced-order model may be obtained by effecting low-rank perturbations to the state-space representation of the original model. The restarts here refer to removing, via oblique projections, some undesirable properties of the reduced-order model, obtained by the Arnoldi process.

We first present this type of implicitly restarted Arnoldi method and then show how this can be applied to obtain a reduced-order model. Finally, we show how to obtain a stable realization from such a reduced-order model.

An Implicitly Restarted Arnoldi Method (Modified Gram-Schmidt)

Suppose that m steps of the Arnoldi algorithm (Algorithm 15.5.10) have been taken, producing a reduced-order model, which has some undesirable features. The question is now how to modify this model by applying an additional oblique projection process so that the undesirable part of the model can be extracted. We state some ideas below due to Jaimoukha and Kasenally (1997) for doing this. Similar ideas were proposed earlier by Gallivan, Grimme and Van Dooren (1994).

Let $T_L, T_R \in \mathbb{R}^{m \times r}$ be the two full-column rank matrices such that $T_L^T T_R = I_r$, an identity matrix of order r . Then the following result by Jaimoukha and Kasenally (1997) shows that the projectors T_L and T_R applied to the reduced-order model may be combined with the oblique projections generated by the Arnoldi process.

Furthermore, it says that the composite projector can be used to obtain a reduced-order model that has the same structure as of the one obtained by Algorithm 15.5.10.

Theorem 15.5.7 Suppose that m steps of the Arnoldi process have been taken obtaining V_m and W_m and let $T_m = W_m^T V_m$ be nonsingular. Let $T_R = Q_R R_R$ and $(T_m)^{-1} T_L = Q_L R_L$ be the QR decomposition of two full column rank matrices T_R and $(T_m^{-1}) T_L$; where $Q_L, Q_R \in \mathbb{R}^{m \times r}$ are parts of orthogonal matrices and $R_L, R_R \in \mathbb{R}^{r \times r}$. Define $V_r = V_m Q_R$, $W_r = W_m Q_L$, and $T_r = W_r^T V_r$. Then

- (i) T_r is nonsingular and $T_r^{-1} = (Q_L^T T_m, Q_R)^{-1} = R_R R_L^T$
- (ii) The triplet (A_r, b_r, c_r) is a reduced-model, where $A_r = T_r^{-1} W_r^T A V_r$, $b_r = T_r^{-1} W_r^T b$, and $c_r = c V_r$.

In the following, we first show how to obtain an implicit Arnoldi scheme (**Algorithm 15.5.11**) for the pairs (A, b) and (A^T, c^T) , and then show how such an implicit scheme can be used for model reduction.

The implicitly restarted Arnoldi method to be described here is a process which augments the existing data from dimension r to dimension m ($2r < m$).

To implement the process, the first r columns of $(H_m, \hat{H}_m^T)^T$ and the first $2r + 1$ columns of (V_m, \tilde{V}_m) should be available prior to restart.

These columns can be obtained as follows:

Let (Q_R, Q_{R_1}) be an $m \times m$ orthogonal matrix such that Q_{R_1} is the orthogonal completion of Q_R (see **Chapter 3**). Suppose that m steps of the Arnoldi method has been taken obtaining l_m and H_m .

Consider now the QR decomposition

$$\begin{pmatrix} Q_{R_1}^T l_m & Q_{R_1}^T H_m Q_R \\ 0 & \tilde{H}_m Q_R \end{pmatrix} = Q(\tilde{l}_r, \tilde{H}_r)$$

where $(\tilde{l}_r, \tilde{H}_r) \in \mathbb{R}^{(r+1) \times (r+1)}$ is upper triangular and Q is a part of an orthogonal matrix.

Define $l_r = Q_R^T l_m$, $\tilde{V}_r = (V_m Q_{R_1}, \tilde{V}_m) Q \in \mathbb{R}^{n \times (r+1)}$, $V_r = V_m Q_R$ and $H_r = Q_R^T H_m Q_R$

We can then write

$$b = V_m l_m = V_m (Q_R, Q_{R_1}) (Q_R, Q_{R_1})^T l_m = V_r l_r + \tilde{V}_r \tilde{l}_r = (V_r, \tilde{V}_r) \begin{pmatrix} l_r \\ \tilde{l}_r \end{pmatrix}$$

Similarly, post-multiplying $AV_m = V_m H_m + \tilde{V}_m \tilde{H}_m$ by (Q_R, Q_{R_1}) , we obtain

$$AV_r = V_r H_r + \tilde{V}_r \tilde{H}_r = (V_r, \tilde{V}_r) \begin{pmatrix} H_r \\ \tilde{H}_r \end{pmatrix}.$$

For example, when $r = 2$ and $m = 6$, we have

$$b = (v_1, v_2, ; v_3, v_4, v_5) \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_3 \\ 0 \\ 0 \end{pmatrix} = (V_2; \tilde{V}_2) \begin{pmatrix} l_2 \\ \tilde{l}_2 \end{pmatrix}$$

$$AV_2 = A(v_1, v_2) = (v_1, v_2) H_2 + (v_3, v_4, v_5) \tilde{H}_2$$

$$\begin{aligned} &= (v_1, v_2; v_3, v_4, v_5) \begin{pmatrix} * & * \\ * & * \\ \cdots & \cdots \\ * & * \\ * & * \\ 0 & * \end{pmatrix} \\ &= (V_2, \tilde{V}_2) \begin{pmatrix} H_2 \\ \tilde{H}_2 \end{pmatrix}. \end{aligned}$$

The above decompositions of b and AV_r show that Arnoldi-like structures are preserved (except for the second term on the right hand of the decomposition of b) even with the application of an additional oblique projection process.

Thus, the terms $(l_r, \tilde{l}_r)^T$, (V_r, \tilde{V}_r) and $(H_r, \tilde{H}_r)^T$ can be used to restart the Arnoldi process with a view to improving the approximation.

Algorithm 15.5.11 *An Implicitly Restarted Arnoldi Algorithm*

Inputs: A - An $n \times n$ matrix
 b - An n -vector
 m - An integer less than n .
 r - An integer such that $2r < m$.

Outputs: $l_m, (H_m^T, \tilde{H}_m^T)^T$ and (V_m, \tilde{V}_m) .

Step 0: Obtain (V_r, \tilde{V}_r) , (H_r, \tilde{H}_r) and (l_r, \tilde{l}_r) using the above QR factorization.

Step 1: Obtain now (V_m, \tilde{V}_m) , which is a part of an orthogonal matrix, and the $(r+1)$ upper Hessenberg matrix $(H_m, \tilde{H}_m)^T$ (i.e., for $1 \leq j \leq m$, $h_{j+r+2,j} = 0$), as follows,

For $j = r+1, r+2, \dots, m$ do

$$w = Av_j$$

For $i = 1, 2, \dots, r+j$ do

$$h_{ij} = v_i^T w$$

$$w = w - v_i h_{ij}$$

End

$$h_{j+r+1,j} \equiv \|w\|_2$$

$$v_{j+r+1} = w/h_{j+r+1,j}$$

End

Step 2: Compute $l_m = (l_r, \tilde{l}_r^T, 0)^T$.

Remark: Observe that $H_m = V_m^T A V_m$ and furthermore the controllability-Arnoldi equations remain valid with $H_m, \tilde{H}_m, V_m, \tilde{V}_m$ and l_m .

A restarted algorithm can also be effected with the observability-Arnoldi equations, by considering the QR decomposition:

$$\begin{pmatrix} Q_{L_1}^T k_m^T & Q_{L_1}^T G_m^T Q_L \\ 0 & G_m^T Q_L \end{pmatrix} = U(\tilde{k}_r^T, \tilde{G}_r^T),$$

where $(\tilde{k}_r^T, \tilde{G}_r^T) \in \mathbb{R}^{(r+1) \times (r+1)}$ is upper triangular and $U \in \mathbb{R}^{(m-r+1) \times (r+1)}$ is part of an orthogonal matrix.

Define now

$$k_r = Q_L^T k_m^T, \quad \tilde{W}_r = (W_m Q_{L_1}, \tilde{W}_m) U \in \mathbb{R}^{n \times (r+1)}, \quad W_r = W_m Q_L, \quad \text{and} \quad G_r = Q_L^T G_m Q_L.$$

Then we can write

$$c^T = W_r k_r^T + \tilde{W}_r \tilde{k}_r^T$$

and

$$A^T W_r = W_r G_r^T + \tilde{W}_r \tilde{G}_r^T.$$

The restarted Arnoldi algorithm can now be applied to augment the above observability-Arnoldi equations and to produce $\hat{G}_m, W_m \tilde{W}_m$ and k_m , satisfying the observability-Arnoldi equations. A model reduction algorithm using implicitly restarted Arnoldi method then can be stated as follows:

Algorithm 15.5.12 An Implicitly Restarted Arnoldi Algorithm for Model Reduction

Inputs: A - The $n \times n$ state-matrix
 b, c - Input and output vectors
 m, r - Integers such that $m > 2r$.

Outputs: $H_m, \tilde{H}_m, V_m, \tilde{V}_m, G_m, \tilde{G}_m, W_m, \tilde{W}_m, l_m$ and k_m .

Step 1. Do m steps of the Arnoldi method with (A, b) to obtain $H_m, \tilde{H}_m, V_m, \tilde{V}_m$ and l_m .

Step 2. Do m steps of the Arnoldi method with (A^T, c^T) to obtain $G_m, \tilde{G}_m, W_m, \tilde{W}_m$ and k_m .

- **Restart:** Do the QR decompositions by choosing the two full-column rank matrices T_L and $T_R \in \mathbb{R}^{m \times r}$ such that $T_L^T T_R = I_r$:

$$T_R = (Q_R, Q_{R_1}) \begin{pmatrix} R_R \\ 0 \end{pmatrix}$$

and

$$(T_m)^{-1} T_L = (Q_L, Q_{L_1}) \begin{pmatrix} R_L \\ 0 \end{pmatrix},$$

where $T_m = W_m^T V_m$.

1. Define the quantities $[l_r, \tilde{l}_r]$, $[V_r, \tilde{V}_r]$, $[H_r, \tilde{H}_r]$; and $[k_r, \tilde{k}_r]$, $[G_r, \tilde{G}_r]$ and $[W_r, \tilde{W}_r]$ by computing the QR decompositions (as shown above):

$$\begin{pmatrix} Q_{R_1}^T l_m & Q_{R_1}^T H_m Q_R \\ 0 & \tilde{H}_m Q_R \end{pmatrix} = Q(\tilde{l}_r, \tilde{H}_r)$$

and

$$\begin{pmatrix} Q_{L_1}^T k_m^T & Q_{L_1}^T G_m^T Q_L \\ 0 & \tilde{G}_m^T Q_L \end{pmatrix} = U(\tilde{k}_r^T, \tilde{G}_r^T),$$

2. Evaluate the residual errors (see below). If satisfied, form the reduced-order model $\{A_r, b_r, c_r\}$ or $(\tilde{A}_r, \tilde{b}_r, \tilde{c}_r)$ as follows and stop; otherwise continue.

Compute

$$T_r^{-1} = R_R R_L^T.$$

- (i) $A_r = T_r^{-1} W_r^T A V_r$, $b_r = T_r^{-1} W_r^T b$ and $c_r = c V_r$,

$$(ii) \quad \tilde{A}_r = W_r^T A V_r T_r^{-1}, \quad \tilde{b}_r = W_r^T b, \quad \tilde{c}_r = c V_r T_r^{-1}.$$

Step 3. Do $m - r$ implicitly restarted steps of the Arnoldi algorithm (**Algorithm 15.5.11**) using $[l_r^T, \tilde{l}_r^T]$, $[H_r^T, \tilde{H}_r^T]$ and $[V_r, \tilde{V}_r]$ to yield $l_m, [H_m, \tilde{H}_m^T]^T$ and $[V_m, \tilde{V}_m]$.

Step 4. Do $m - r$ implicitly restarted steps of the Arnoldi algorithm using $[k_r, \tilde{k}_r]$, $[G_r, \tilde{G}_r]$ and $[W_r, \tilde{W}_r]$ to yield $k_m, [G_m, \tilde{G}_m]$ and $[W_m, \tilde{W}_m]$.

Residual Errors. The residual errors needed in Step 2 can be computed as follows: (See Jaimoukha and Kasenally (1997) for details):

$$\begin{aligned} \bullet \quad r_b^r &= \left\| b - (sI - A)V_r h_r(s) \right\|_\infty \equiv \left\| \begin{pmatrix} T_r^{-1} W_r^T \tilde{V}_r \\ I \end{pmatrix} (\tilde{l}_r + \tilde{H}_r h_r(s)) \right\|_\infty \\ \bullet \quad r_c^r &= \left\| c - g_r(s)W_r^T(sI - A) \right\|_\infty \equiv \left\| (\tilde{k}_r + g_r(s)\tilde{G}_r)(\tilde{W}_r^T V_r T_r^{-1}, \quad I) \right\|_\infty. \end{aligned}$$

The following property of the reduced-order model, obtained by the above algorithm, holds.

For a proof, see Jaimoukha and Kasenally (1997).

Theorem 15.5.8 (Matching of Markov Parameters with Restarted Arnoldi)

Let $T_m = W_m^T V_m$ be nonsingular. Define

$$\begin{aligned} A_m &= H_m + T_m^{-1} W_m^T \tilde{V}_m \tilde{H}_m = T_m^{-1} W_m^T A V_m \\ b_m &= l_m = T_m^{-1} W_m^T b \\ c_m &= k_m T_m = c V_m \end{aligned}$$

where $W_m, V_m, H_m, \tilde{H}_m, l_m, k_m, G_m$ and \tilde{G}_m are obtained by Algorithm 15.5.12.

Let k be the largest integer satisfying $k \leq \frac{m}{r+1}$.

Then

$$A^{i-1} b = V_m A_m^{i-1} b_m, \quad 1 \leq i \leq k.$$

and

$$c A^{i-1} b = c_m A_m^{i-1} b_m, \quad 1 \leq i \leq 2k.$$

Remarks: Note that $r = 0$ corresponds to the Arnoldi method without restart. Thus the restarted method matches a fewer Markov parameters than that obtained by no restart. However, it replicates the low-frequency characteristic of the original model, while without restart, the reduced-order model tends to replicate the high-frequency characteristics.

Example 15.5.7 was run again with Algorithm 15.5.12 with $m = 4, r = 1$. The following table shows that Markov parameters for the original and reduced-order match, as predicted in

Theorem 15.5.8.

Table 15.5 (Matching of Markov parameter with Implicit Restart).

i	$cA^{i-1}b$	$c_m A_m^{i-1} b_m$
1	6.123204	6.123204
2	-1.568586	-1.568586
3	89.833149	89.833149
4	-401.858937	-401.858937

Extracting Stable Part of the Reduced-order Model and Balanced Truncation

Obtaining T_L and T_R : The matrices T_L and T_R needed in the restart step should be chosen so that some undesirable properties can be extracted with the restart process. Below we show how to choose them to extract (i) the unstable part and then (ii) undesirable redundant modes. As said before, the reduced-order models obtained by the Arnoldi methods may be unstable. We have seen in Chapter 14 that an ordered Schur method can be used to extract the stable part of the reduced-order model. We repeat this procedure here.

Let (A_m, b_m, c_m) be an unstable reduced-order model.

Let

$$T_1 A_m T_1^T \equiv A_s = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

be a block ordered-real Schur form of A_m ; where T_1 is orthogonal, $A_{11} \in \mathbb{R}^{p \times p}$ is stable, and $A_{22} \in \mathbb{R}^{(m-p) \times (m-p)}$ is unstable. The next step is to eliminate A_{12} . This can be done using a solution of Sylvester equation (see Chapter 14).

Let X_1 be a solution of the Sylvester equation

$$A_{11}X - XA_{22} + A_{12} = 0.$$

Define now $T_{L_1} = T_1^T \begin{pmatrix} I \\ -X_1^T \end{pmatrix}$ and $T_{R_1} = T_1^T \begin{pmatrix} I \\ 0 \end{pmatrix}$,

and

$$A_{11} = T_{L_1}^T A_m T_{R_1}, \quad b_1 = T_{L_1} b_m, \quad \text{and } c_1 = c_m T_{R_1}.$$

Then it can be shown that the model (A_{11}, b_1, c_1) is stable.

(The matrices T_{L_1} and T_{R_1} now can be used in the restart step of Algorithm 15.5.4 ($T_L \equiv T_{L_1}, T_R \equiv T_{R_1}$), if the purpose is to obtain a stable reduced-order model using implicit Arnoldi scheme).

Extracting the Redundant Modes

Once the stable reduced-order model (A_{11}, b_1, c_1) is obtained using the above method, a square-root type of algorithm (see Chapter 14) can be applied to the above model to remove any redundant modes.

Suppose that the first r modes are to be retained. Then

- Compute the controllability and observability Grammians C_G^S and O_G^S by solving the Lyapunov equations:

$$A_{11}C_G^S + C_G^S A_{11}^T + b_1 b_1^T = 0$$

and

$$A_{11}^T O_G^S + O_G^S A_{11} + c_1^T c_1 = 0.$$

- Find the Cholesky factorizations $C_G^S = L_r L_r^T$ and $O_G^S = L_o L_o^T$ and take the SVD of $L_o^T L_r = \hat{U} \Sigma_p \hat{V}^T$ where $\Sigma_p = \text{diag}(\sigma_1, \dots, \sigma_p)$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p$.

- Define the transformations

$$T_{L_2} = L_o \hat{U}_r \Sigma_r^{-\frac{1}{2}} \in \mathbb{R}^{r \times p}$$

and

$$T_{R_2} = L_r \hat{V}_r \Sigma_r^{-\frac{1}{2}} \in \mathbb{R}^{r \times p},$$

where $\Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r)$ and \hat{U}_r and \hat{V}_r are the first r columns of \hat{U} and \hat{V} .

- Then the required reduced-order model is given by:

$$A_r = T_{L_2}^T A_{11} T_{R_2}, \quad b_r = T_{L_2}^T b_1 \quad \text{and} \quad c_r = c_1 T_{R_2}.$$

The matrices T_{L_2} and T_{R_2} can now be used in the restart step of Algorithm 15.5.12, if the purpose is to obtain a reduced-order model that does not contain any redundant modes.

Combining T_L and T_R for the Implicit Arnoldi Scheme.

In practice, the reduced-order model (A_m, b_m, c_m) obtained by the Arnoldi algorithm may be unstable and also may contain several redundant modes. Then, one would, naturally, like to combine the above two transformations to obtain a reduced-order model that is both stable and does not contain any redundant modes. Such composite projectors are given as follows. Let $T_{L_i}, T_{R_i}, i = 1, 2$ are computed as above. Then compute

$$T_L = T_{L_1} T_{L_2} = T_1^T \begin{pmatrix} I \\ -X_1^T \end{pmatrix} L_o \hat{U}_r \Sigma_r^{-\frac{1}{2}}$$

and

$$T_R = T_{R_1} T_{R_2} = T_1^T \begin{pmatrix} I \\ 0 \end{pmatrix} L_r \hat{V}_r \Sigma_r^{-\frac{1}{2}}.$$

These matrices T_L and T_R can now be applied to (A_m, b_m, c_m) to obtain the reduced-order model (A_r, b_r, c_r) in a single step. *Such reduced-order model will then be stable and not contain any redundant modes.*

The Cross-Grammian Approach to Model Reduction - A Restarted Arnoldi Method

Another implicitly restarted Arnoldi method closely related to the one discussed in the last section is the Cross Grammian method, recently proposed by Antoulas, Sorensen and Gugercin (2001).

The **Cross Grammian** X is defined for square system ($m = r$) as the solution to the Sylvester equation

$$AX + XA + BC = 0.$$

It can be shown that in SISO case the solution X is similar to a symmetric matrix and the cross Grammian X is related to the controllability and observability Grammians C_G and O_G by the relation:

$$X^2 = C_G O_G.$$

The approach suggested in Sorensen and Antoulas (2001) consists in constructing low rank- k approximate solutions by setting $X = V\hat{X}W^T$, where $W^T V = I_k$ and then projecting using V and W together.

An implicitly started Arnoldi method has been developed to compute an approximation X_k to the best rank- k approximation to X .

For details of the Arnoldi method to compute X_k , the readers are referred to the above paper. We show here only how to obtain a reduced-order model once X_k is constructed.

In the SISO case, (that is, when B and C are vectors b and c , respectively), the matrix X has real eigenvalues. Then, if X_k were obtained as $X_k = Z_k D_k W_k^T$, where $W_k^T Z_k = I$ and the diagonal entries of D_k are the dominant Hankel singular values, then the matrices Z_k and W_k would have provided a balanced realization. However, X_k constructed by the implicit Arnoldi method gives only the best rank k approximation to X . Thus, it is necessary to approximate the relevant eigenvectors basis for X with an eigenvector basis for X_k .

Let $X_k = U_k S_k V_k^T$ be an approximation to the best rank- k approximation to X .

Define $G = S_k^{1/2} V_k^T U_k S_k^{1/2}$.

If $GZ = ZD_k$ with D_k real and diagonal, then taking

$$Z_k = U_k S_k^{1/2} Z |D_k|^{-1/2}$$

and

$$W_k = V_k S_k^{1/2} Z^{-T} |D_k|^{-1/2},$$

We have $X_k = Z_k D_k W_k^T$ with $W_k^T Z_k = I_k$.

Furthermore,

$$XZ_k = X_k Z_k + (X - X_k) Z_k = Z_k D_k + O(\sigma_{k+1})$$

Thus, the matrices Z_k and W_k can be used to construct an approximate balanced realization. The reduced-order model is now obtained as

$$A_k = W_k^T A Z_k, \quad b_k = W_k^T b, \quad c_k = c Z_k.$$

Extension to the MIMO Case.

The crucial property in the SISO case is $X^2 = C_G O_G$.

This relationship also holds true for a MIMO system which is symmetric; that is for a system whose transfer function is symmetric. In MIMO case, the solution matrix X of the Sylvester equation $AX + XA + BC = 0$ can have complex eigenvalues and it is then more appropriate to use the real Schur form as eigendecomposition.

Let $S_k^{1/2}(V_k^T U_k)S_k^{1/2}\hat{Z} = \hat{Z}R_k$ be a real Schur form, where $\hat{Z}^T\hat{Z} = I_k$ and R_k is a real Schur matrix.

Define now

$$Z_k = U_k S_k^{1/2} \hat{Z} R_k^{-1}, \quad W_k = V_k S_k^{1/2} \hat{Z}.$$

Then $W_k^T Z_k = I_k$, $X_k Z_k = Z_k R_k$ and $W_k^T X_k = R_k W_k^T$.

Thus, again the reduced-order model (A_k, B_k, C_k) can be obtained as:

$$A_k = W_k^T A Z_k, \quad B_k = W_k^T B \text{ and } C_k = C Z_k.$$

Remarks on Stability and Balancing

The Cross-Grammian method presented above may not give a stable reduced-order system, even if the original system is stable.

The implicitly restarted technique of Grimme, Van Dooren and Sorensen (1996), stated in Section 15.5.11, can be applied to the projected quantities to restore the stability.

The reduced-order model obtained above may also not be balanced; it is only approximately balanced.

15.6 Summary and Review

The standard numerical methods for control, although viable for dense problems, are not suitable for large and sparse problems. This is because, like most large practical problems, large control problems, such as those arising in the design and analysis of structures like buildings, bridges, air and space crafts, power systems, computer networks, etc., are sparse; and the sparsity is completely destroyed by use of these standard methods. This is because, these control algorithms are based on canonical forms, such as Hessenberg, real-Schur, controller-Hessenberg, observer-Hessenberg, etc., which are achieved by using numerical linear algebra techniques like Gaussian elimination, Householder's and Givens' transformations, QR iteration, QZ iterations, etc., and these techniques are not capable of preserving the sparsity in a problem.

In the last few years, attempt has been made to solve large and sparse control problems by using the Krylov subspace methods, which have been found to be effective in solving large-scale matrix problems. These methods are based on matrix-vector multiplications and have the ability to preserve the sparsity.

A brief account of some of these Krylov subspace methods for control has been given in this Chapter.

I. Arnoldi and Lanczos Methods for Controllability and Observability

Running Arnoldi or Lanczos method with (A, B) and (A^T, C^T) , one can compute, respectively, orthonormal bases of the controllable, observable, uncontrollable and unobservable subspaces for the linear system (A, B, C) . These orthonormal bases then can be used to compute the transforming matrix T to compute the Kalman decomposition of the system (A, B, C) . Details are given in **Section 15.5.2**.

II. Arnoldi Methods for Lyapunov Equations

Consider the single-input Lyapunov equation:

$$AX + XA^T + bb^T = 0,$$

with A as a stable matrix.

By running m -steps of the Arnoldi method with $v_1 = \frac{b}{\|b\|}$, one obtains an $m \times m$ Hessenberg matrix H_m and an orthonormal matrix V_m , which gives rise to the projected equation:

$$H_m G_m + G_m H_m^T + \beta^2 e_1 e_1^T = 0.$$

This small equation can now be solved for the matrix G_m using the Schur method, described in **Chapter 8** and then an approximation X_m to the solution X can be obtained as

$$X_m = V_m G_m V_m^T.$$

Galerkin condition: It can be shown that the residual matrix $Res(X_m) = AX_m + X_m A^T + bb^T$ satisfies the Galerkin property: $V_m^T Res(X_m) V_m = 0$.

Furthermore, the residual norm error can be computed cheaply, which can be used to **restart the Arnoldi** method, if this error is not satisfactory. For details, see **Section 15.5.3**.

In the multi-input case of the Lyapunov equation: $AX + XA^T + BB^T = 0$, the block Arnoldi method can be used in a similar fashion. A block Arnoldi method for discrete Lyapunov equation: $AXA^T - X + BB^T = 0$ can also be developed in the same way (see **Algorithm 15.5.3**). This block Arnoldi method again satisfies the Galerkin property.

III. Arnoldi Methods for Sylvester Equation

It is well-known (see **Chapter 8**) that solving the Sylvester equation

$$AX - XB = C$$

is equivalent to solving the algebraic system

$$(I \otimes A - B^T \otimes I)x = c.$$

Since it is computationally prohibitive to solve the above system with large and sparse matrices A and B , the Arnoldi method can be used to project the system to a smaller system and then an approximate solution of the original system can be obtained from the solution of the projected smaller system. The residual vector at each iteration step can be computed cheaply and used to

restart the method. A restarted Arnoldi method for Sylvester equation is given in **Algorithm 15.5.4**.

For details, see **Section 15.3.2** and **15.5.6**.

Assuming that the matrix B is of order $n \times p$, where $p \ll n$, block Arnoldi methods can be developed to solve the Sylvester equation $AX - XB = C$. Two such algorithms, one the Galerkin-type and the other GMRES-type have been described in **Section 15.5.7**.

IV. Arnoldi Method for Sylvester-observer Equation

Recall that the single-output Sylvester-observer equation arising in reduced-order state-estimation is

$$A^T X - XH = (0, c),$$

where A and c are given and X and H have to be found in such a way that $\Omega(H)$ is a given set of numbers $\{\mu_1, \dots, \mu_m\}$. An Arnoldi method for the above equation is developed (**Section 15.5.8**) by observing the striking resemblance of this equation with the **Arnoldi equation**, obtained after the m steps of running the Arnoldi method on (A^T, v_1) :

$$A^T V_m - V_m H_m = h_{m+1,m}(0, v_{m+1}).$$

This resemblance leads us to the problem of choosing the vector v_1 appropriately so that the last vector v_{m+1} , obtained after m steps of the Arnoldi methods becomes the output vector c . It is shown that this will happen if v_1 is chosen as the normalized solution x of the algebraic system:

$$q(A^T)x = c^T,$$

where $q(t) = (t - \mu_1)(t - \mu_2) \cdots (t - \mu_m)$. This choice of v_1 will give the vector c , as desired, but will not guarantee that the matrix H_m has the spectrum $\{\mu_1, \dots, \mu_m\}$.

For this to happen, an eigenvalue assignment algorithm, such as **Algorithm 11.2.1** is to be used to assign the spectrum of H_m to the set $\{\mu_1, \dots, \mu_m\}$.

An approximate solution X_m is then obtained as

$$X_m = \left(\frac{1}{\alpha} \right) V_m.$$

Here α is the scaling factor determined by the last column of $A^T V_m - V_m H_m$.

A partial fraction approach is described to solve the above large-scale system. For details, see **Section 15.5.8**.

V. Arnoldi Method the Algebraic Riccati Equation

A brief sketch of an Arnoldi method for the CARE: $XA + A^T X - XBB^T X + LL^T = 0$ is given in **Section 15.5.9**.

Let m steps of the block Arnoldi method is run on A with the starting block vector V_1 obtained from the QR factorization of L : $L = V_1 R$, producing H_m , U_m and L_m .

Define $B_m = U_m^T B$ and L_m by $U_m L_m = L$. Then a low-rank solution X_m of the solution X of the CARE is given by

$$X_m = U_m G_m U_m^T,$$

where G_m satisfies the projected CARE:

$$G_m H_m + H_m^T G_m - G_m B_m B_m^T G_m + L_m L_m^T = 0.$$

It can be shown that the residual norm $\text{Res}(X_m)$ satisfies the Galerkin condition.

VI. Arnoldi Method for Partial Pole-Placement:

The partial pole-placement problem is the problem of reassigning a small number of eigenvalues of a matrix A by suitable feedback, while keeping the rest of the spectrum unchanged.

A projection algorithm, based on partial Schur factorization of A^T (which can be implemented using the Arnoldi method) has been described in **Section 15.5.10**. Let $\Omega(A) = \{\lambda_1, \dots, \lambda_k; \lambda_{k+1}, \dots, \lambda_n\}$.

Let

$$A^T Q = Q R$$

be the partial Schur factorization of A^T , where the $k \times k$ quasi-triangular matrix R contain the eigenvalues $\lambda_1, \dots, \lambda_k$ to be reassigned.

Suppose G^T is the feedback matrix solving the projected problem; that is, G^T is such that $\Omega(R^T - S_0 G^T) = \{\mu_1, \dots, \mu_k\}$, where $S_0 = B^T Q$ and $\{\mu_1, \dots, \mu_k\}$ is the set replacing the set $\{\lambda_1, \dots, \lambda_k\}$.

Then it can be shown (**Theorem 15.5.3**) that the feedback matrix $K = (QG)^T$ solves the partial eigenvalue assignment problem; that is, $\Omega(A - BK) = \{\mu_1, \dots, \mu_k; \lambda_{k+1}, \dots, \lambda_n\}$.

VII. Lanczos and Arnoldi Methods for Model Reduction

Because of the immense importance of the model reduction problem, not only in control, but also in the other application areas, the model reduction problem for large and sparse system has been very well-studied and several Lanczos and Arnoldi method have been developed in recent years.

We have described a Lanczos Algorithm (**Algorithm 15.5.9**), An Arnoldi Algorithm (**Algorithm 15.5.10**), and an implicitly restarted Arnoldi Algorithm (**Algorithm 15.5.12**) for model reduction. A basic idea behind all these algorithms is the same: The Arnoldi and Lanczos methods are used to project the problem into an m -dimensional Krylov subspace and a reduced-order model is obtained thereby.

Algorithm 15.5.9 and **Algorithm 15.5.10** are capable of producing a reduced-order model for the SISO problem (A, b, c) such that the first few moments of both the original and the reduced-order models are the same. Thus, if (A_m, b_m, c_m) is a reduced-order model obtained after m -steps of the Lanczos method, then it can be shown (**Theorem 15.5.4**) that

$$c A^{i-1} b = c_m A_m^{i-1} b_m, \quad i = 1, 2, \dots, 2k.$$

Similar results hold for the Arnoldi Algorithm (**Algorithm 15.5.10**).

Unfortunately, **these algorithms can not guarantee the stability of the reduced-order model**, even if the original model is stable.

To guarantee stability or to achieve other properties from the reduced-order model, an **implicit scheme** needs to be used along with the original method.

Such an **implicit scheme** has been described for Lanczos method in **Section 15.5.11**.

An implicit scheme (in fact, an implicit restarted scheme) for Arnoldi model reduction method (**Algorithm 15.5.12**) has also been described in this section.

Algorithm 15.5.12 is of particular importance. It can be used to obtain a reduced-order model that is **stable and free of any redundant modes**.

Finally, in this Section, a recent new approach for model reduction, based on **Cross Grammian**, which is defined to be the solution of the Sylvester equation

$$AX + XA + BC = 0$$

for the model (A, B, C) , is described briefly.

An **Arnoldi method** can be used to compute the best rank- k approximation X_k to the solution X of the above equation and once such an approximation is obtained, a reduced-order model can be constructed from the SVD of X_k . The method for obtaining X_k is not provided in this Section, but it is shown how to compute the reduced-order model from the SVD of X_k .

This Cross-Grammian approach is still in developing stage and further research is underway.

It is worth mentioning that research on Krylov subspace methods for control problems itself, is still in its infancy and much remains to be done.

15.7 Chapter Notes and Further Reading

In this chapter, we have provided a brief review of some of the existing Krylov subspace methods for a few large problems arising in design and analysis of control problems. These include Lanczos and Arnoldi methods for determining controllability and observability by Boley (1994) and Boley and Golub (1984, 1991); Arnoldi methods for Lyapunov and Sylvester equations by Saad (1990), Hu and Reichel (1992), Jaimoukha and Kasenally (1994); Arnoldi method for the single-output Sylvester-observer equation by Datta and Saad (1991); a projection algorithm (which can be implemented using Arnoldi method) for partial eigenvalue assignment problem by Saad (1988); and Lanczos and Arnoldi methods for model reduction by Boley (1994), Grimme, Sorensen and Van Dooren (1996), Jaimoukha and Kasenally (1994, 1995, 1997) and Sorensen and Antoulas (2001).

The Hu-Reichel algorithm was extended by Simoncini (1996) to block form. There also have been some recent developments on the Krylov subspace methods for Sylvester equation. El Guennouni, Jbilou and Riquet (2000) have developed block Arnoldi and nonsymmetric block Lanczos algorithms for Sylvester equation. Robb   and Sadkane (2002) have proposed new block

Arnoldi and block GMRES methods for Sylvester equation and analyzed their convergence properties in details.

In the context of model reduction, it is noted that there are other important methods, such as the Padé via Lanczos (PVL), the interpolation methods, etc., which have not been included here. For details of these methods, the readers are referred to the associated papers cited in the Reference Section of this Chapter. In particular, for Lanczos methods of model reduction see, Feldman and Freund (1995a, 1995b, and 1995c), Jaimoukha and Kasenally (1995), Grimme, Sorensen and Van Dooren (1996), Papakos and Jaimoukha (2001), Papakos (2001), Papakos and Jaimoukha (2002), Gallivan, Grimme and Van Dooren (1996), etc. The papers by Papakos and Jaimoukha (2002) contains a procedure for model reduction combining nonsymmetric Lanczos algorithm and Linear Fractional Transformations (LFT). The delightful recent surveys by Freund (1999), the recent research monograph by Antoulas (2002), Ph.D. thesis by Grimme (1994), and short course lecture notes by Van Dooren (1995, 2000) and Feldman and Freund (1995) are good sources of knowledge for model reduction. The paper by Freund (1999) includes 123 references on large-scale matrix computations using Krylov methods and their applications to model reduction. The earlier general surveys on Krylov subspace methods in control include the papers by Boley (1994), Datta (1997), Boley and Datta (1996), Van Dooren (1992), Bultheel and Van Barel (1986), and Fortuna et al. (1992). Some other papers of interest on Krylov subspace methods for model reduction include the papers by Villemagne and Skelton (1987), and Su and Craig, Jr. (1991).

For recent results on partial eigenvalue and eigenstructure assignments, which include results on conditions of existence and uniqueness of solutions and Sylvester equation based algorithms for both these problems, see Sarkissian (2001) and Datta and Sarkissian (2002).

EXERCISES

1. Work out a modified version of Algorithm 15.3.4 incorporating column pivoting in case V does not have full rank.
2. Establish the Galerkin property of the residual and the residual error norm for the approximate solution X_m of the Lyapunov equation $AX + XA^T + bb^T = 0$, obtained by the Arnoldi method (**Algorithm 15.5.1**).
3. Establish the perturbation result for Algorithm 15.5.2.
4. Develop a block-Arnoldi method for the multi-input Lyapunov equation: $AX + XA^T + BB^T = 0$, analogous to Algorithm 15.5.1 and establish the Galerkin property of the residual and the residual error norm of the projected solution.
5. Develop an explicitly restarted block Arnoldi method for the Lyapunov equation $AXA^T - X = -BB^T$ and obtain a perturbation result.
6. Develop a GMRES-type algorithm for the discrete-time Lyapunov equation $AXA^T - X = -BB^T$.
7. Fill in the details of the minimal projected residual algorithm started in Section 15.5.7.
8. Develop an explicitly restarted-Arnoldi method for the discrete Lyapunov equation $AXA^T - X = -BB^T$.
9. Prove that the $Res(X_m)$ obtained by Algorithm 15.5.3 satisfies a Galerkin property (consult Jaimoukha and Kasenally (1994)).
10. Develop an explicitly restarted-Arnoldi method for the continuous-time Algebraic Riccati equation: $XA + A^TX - XSX + Q = 0$.
11. Prove that in the case of happy breakdown in the Arnoldi method, the block Arnoldi (block version of Algorithm 15.5.1) and GMRES (Algorithm 15.5.4) methods for the Lyapunov equation $AX + XA^T + BB^T = 0$ yield the same solution, namely, the exact solution X .
12. Write the details of implementation of Algorithm 15.5.8 (**Partial pole-placement by projection**) using the block Arnoldi algorithm (**Algorithm 15.3.5**).
13. Write the restarted Arnoldi method to find a reduced-order model that is stable and free of redundant modes in details.
14. Develop a restarted GMRES algorithm for solving the Sylvester equation $AX - XB = C$ (Consult Hu and Reichel (1992), and Robb   and Sadkane (2002)).
15. Give a proof of Theorem 15.5.3.

16. Give the details of how to solve the least-squares problem arising in the GMRES-type method for the solution of the Sylvester equation $AX - XB = C$ in **Algorithm 15.5.1**.

Research Problems on Chapter 15

1. Develop a projection algorithm of the block-Arnoldi type to solve the multi-output Sylvester-observer equation $AX - XB = GC$, analogous to the Datta-Saad algorithm (Algorithm 15.5.6) for the single-output problem.
2. Develop a projection algorithm for the discrete Algebraic Riccati equation (DARE):
$$A^T X A - X + Q - A^T X B (R + B^T X B)^{-1} B^T X A = 0.$$
3. Develop a block-Arnoldi algorithm to compute the frequency response matrix $G(j\omega) = D + C(\omega I - A)^{-1}B$.
4. Develop an Arnoldi or Lanczos based algorithm for the model identification problem.

REFERENCES

1. J.I. Aliaga, D.L. Boley, R.W. Freund, and V. Hernández, A Lanczos-type method for multiple starting vectors, *Math. Comp.*, vol. 69, pp. 1577-1601, 2000.
2. A.C. Antoulas, D.C. Sorensen and S. Gugercin, A Survey of model reduction methods for large-scale systems, *Contemporary Mathematics*, American Mathematical Society, Providence, vol. 280, pp. 193-219, 2001, (V. Olshevsky, Editor).
3. A.C. Antoulas, *Lectures on Approximations of Large-Scale Dynamical Systems*, SIAM, Philadelphia, 2003 (To appear).
4. W.E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, *Quart. Appl. Math.*, vol. 9, pp. 17-29, 1951.
5. O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, New York, 1994.
6. Z. Bai, P. Feldmann, and R.W. Freund, Stable and passive reduced-order models based on partial Padé approximation via the Lanczos process, *Numerical Analysis Manuscript*, No. 97-3-10, Bell Laboratories, Murray Hill, NJ, 1997.
7. Z. Bai, D. Day and Q. Ye, ABLE: An adaptive block Lanczos method for nonhermitian eigenvalue problem, *SIAM J. Matrix Anal. Appl.*, pp. 1060-1082, 1999.
8. Z. Bai and R. Freund, A band symmetric Lanczos process based on coupled recurrences with applications in model-order reduction, *Numerical Analysis Manuscript*, Bell Laboratories, Murray Hill, NJ, 1999.
9. Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. Van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems, A Practical Guide*, SIAM, Philadelphia, 2000.
10. R. Barret, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.
11. D.L. Boley, Krylov space methods on state-space control models, *Proc. Circ. Syst. & Signal*, vol. 13, pp. 733-758, 1994.
12. D. Boley and B.N. Datta, Numerical methods for linear control systems, *Systems and Control in the Twenty-First Century*, Birkhauser, Boston, pp. 51-74, 1996, (C. Byrnes, B. Datta, D. Gilliam, and C. Martin, Editors)
13. D. Boley and G. Golub, The nonsymmetric Lanczos algorithm and controllability, *Syst. Contr. Lett.*, vol. 16, pp. 97-105, 1991.
14. D. Boley and G. Golub, The Lanczos-Arnoldi algorithm and controllability, *Syst. Contr. Lett.*, vol. 4, pp. 317-327, 1984.

15. C. Brezinski, M. Redivo Zaglia, and H. Sadok, Avoiding breakdown and near-breakdown in Lanczos type algorithms, *Numer. Algorithms*, vol. 1, pp. 26-284, 1991.
16. A. Bultheel and M. Van Barel, Padé techniques for model reduction in linear systems theory; a survey, *J. Comput. Appl. Math.*, vol. 14, pp. 401-438, 1986.
17. D. Calvetti and L. Reichel, Numerical aspects of solution methods for large Sylvester-like observer-equations, *Proc. IEEE Conf. Dec. Control*, pp. 4389-4397, 1997.
18. D. Calvetti, E. Gallopoulos, and L. Reichel, Incomplete partial fractions for parallel evaluations of rational matrix functions, *J. Comp. Appl. Math.*, vol. 59, pp. 349-380, 1995.
19. D. Calvetti, B. Lewis and L. Reichel, On the solution of large Sylvester-observer equation, *Num. Lin. Alg. Appl.*, vol. 8, pp. 435-451, 2001.
20. D. Calvetti, B. Lewis, and L. Reichel, Partial eigenvalue assignment for large linear control systems, *Contemporary Mathematics*, American Mathematical Society, Providence, RI, vol. 28, pp. 24-254, 2001. (V. Olshevsky, Editor)
21. F. Chatelin, *Eigenvalues of Matrices*, Wiley, New York, 1993.
22. B.N. Datta, An algorithm to assign eigenvalues in a Hesseberg matrix: single-input case, *IEEE Trans. Automat. Control*, AC-32, pp. 414-417, 1987.
23. B.N. Datta, Linear and numerical linear algebra in control theory: Some research problems, *Lin. Alg. Appl.*, vol. 197/198, pp. 755-790, 1994.
24. B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, Pacific Grove, CA, 1995.
25. B.N. Datta, Krylov-subspace Methods for Control: An Overview, *Proc. IEEE Conf. Dec. Control*, 1997.
26. B.N. Datta and C. Hetti, Generalized Arnoldi methods for the Sylvester-observer matrix equation and the multi-input eigenvalue assignment problems, *Proc. IEEE Conf. Dec. Control*, pp. 4379-4383, 1997.
27. B.N. Datta and Y. Saad, Arnoldi methods for large Sylvester-like observer matrix equations, and an associated algorithm for partial spectrum assignment, *Lin. Alg. Appl.*, vol. 154-156, pp. 225-244, 1991.
28. B.N. Datta and D. Sarkissian, Partial eigenvalue assignment: Existence, uniqueness, and numerical solutions, *Proc. Mathematical Theory of Networks and Systems*, (MTNS' 2002), Notre Dame, August, 2002.
29. J.J. Dongarra, I.S. Duff, D.C. Sorensen, and H. A. Van der Vorst, *Numerical Linear Algebra for High-Performance Computers*. SIAM, Philadelphia, PA, 1998.

30. I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, UK, 1986.
31. A. El Guennouni, K. Jbilou and J. Riquet, Block Krylov subspace methods for solving large Sylvester equations, preprint, LMPA, No. 132, Université du Littoral, 2001 (To appear in *Numer. Algorithms*), 2003.
32. P. Feldmann and R.W. Freund, Efficient linear circuit analysis by Padé approximation via the Lanczos process, *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 639-649, 1995a.
33. P. Feldman and R.W. Freund, *Numerical Simulation of Electronic Circuits: State-of-the-Art Techniques and Challenges*, Course Notes, 1995b (*Available on-line from <http://cm.bell-labs.com/who/Freund>*).
34. P. Feldman and R.W. Freund, Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm, *Proc. 32nd Design and Automation Conference, Assoc. Comp. Machinery*, pp. 474-479, 1995c.
35. L. Fortuna, G. Nunnari, and A. Gallo, *Model Order Reduction Techniques with Applications in Electric Engineering*, Springer-Verlag, London, UK, 1992.
36. R.W. Freund, Computation of matrix Padé approximations of transfer function via a Lanczos-type process, *Approx. Theory VIII*, vol. 1, pp. 215-222, 1995.
37. R.W. Freund and N.M. Nachtigal, QMRPACK: *A package of QMR algorithms*, *ACM Trans. Math. Software*, vol. 22, pp. 46-77, 1996.
38. R.W. Freund, Computing minimal partial realization via a Lanczos-type algorithm for multiple starting vectors, *Proc. IEEE Conf. Dec. Control*, pp. 4394-4399, 1997.
39. R.W. Freund, Reduced-order modeling techniques based on Krylov subspace methods and their uses in circuit simulation, *Applied and Computational Control, Signals, and Circuits*, vol. 1, Birkhauser, Boston, pp. 435-498, 1999, (B.N. Datta, et. al., Editors).
40. R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal, An implementation of the look-ahead Lanczos algorithm for non-hermitian matrices, *SIAM J. Sci. Comput.*, vol. 14, pp. 137-158, 1993.
41. R.W. Freund and N.M. Nachtigal, QMR: A quasi-minimal residual method for non-Hermitian linear systems, *Numer. Math.*, vol. 60, pp. 315-339, 1991.
42. R.W. Freund and N.M. Nachtigal, An implementation of the QMR based on coupled two-term recurrences, *SIAM J. Sci. Comput.*, vol. 15, pp. 313-337, 1994.
43. K. Gallivan, E.J. Grimme, and P. Van Dooren, Asymptotic waveform evaluation via a restarted Lanczos method, *Appl. Math. Lett.*, vol. 7, pp. 75-80, 1994.

44. K. Gallivan, E.J. Grimme, and P. Van Dooren, A rational Lanczos algorithm for model reduction, *Numer. Alg.*, vol. 12, pp. 33-63, 1996.
45. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Edition, Johns Hopkins University, Baltimore, MD, 1996.
46. W.B. Gragg, Matrix interpolations and applications of the continued fraction algorithm, *Rocky Mountain J. Math.*, vol. 4, pp. 213-225, 1974.
47. W.B. Gragg and A. Lindquist, On the partial realization problem, *Lin. Alg. Appl.*, vol. 50, pp. 277-319, 1983.
48. E.J. Grimme, D.C. Sorensen and P. Van Dooren, Model reduction of state space systems via an implicitly restarted Lanczos method, *Numer. Alg.*, vol. 1, pp. 1-32, 1996.
49. E.J. Grimme, *Krylov Projection Methods for Model Reduction*, Ph.D. Thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1994.
50. M. Hochbruck and C. Lubich, On Krylov subspace approximations to the matrix exponential operator, *SIAM J. Numer. Anal.*, 1985.
51. D.Y. Hu and L. Reichel, Krylov subspace methods for the Sylvester equation, *Lin. Alg. Appl. Appl.*, vol. 172, pp. 283-313, 1992.
52. J.L. Howland and J.A. Senez, A constructive method for the solution of the stability problem, *Numer. Math.*, vol. 16, pp. 1-7, 1970.
53. I.M. Jaimoukha, A general minimal residual Krylov subspace method for large-scale model reduction, *IEEE Trans. Automat. Control*, vol. 42, pp. 1422-1427, 1997.
54. I.M. Jaimoukha and E.M. Kasenally, Krylov subspace methods for solving large Lyapunov equations, *SIAM J. Numer. Anal.*, vol. 31, pp. 227-251, 1994.
55. I.M. Jaimoukha and E.M. Kasenally, Implicitly restarted Krylov Subspace methods for stable partial realizations, *SIAM J. Matrix Anal. Appl.*, vol. 18, no. 3, pp. 633-652, 1997.
56. I.M. Jaimoukha and E.M. Kasenally, Oblique projection methods for large scale model reduction, *SIAM J. Matrix Anal. Appl.*, vol. 16, pp. 602-627, 1995.
57. C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Nat. Bur. Standards*, vol. 45, pp. 255-282, 1950.
58. R.B. Lehoucq, D.C. Sorensen, and C. Yang, ARPACK Users' Guide: *Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, PA, 1988.

59. V. Papakos and I.M. Jaimoukha, Implicitly restarted Lanczos algorithm for model reduction, *Proc. 40th IEEE Conf. Dec. Control*, pp. 3671-3672, 2001.
60. V. Papakos, *An implicitly restarted nonsymmetric Lanczos algorithm for model reduction*, MPhil to Ph.D. Transfer Report, Imperial College, London, U.K., 2001.
61. V. Papakos and I.M. Jaimoukha, Model reduction via an LFT-based explicitly restarted Lanczos algorithm (preprint), *Proc. Mathematical Theory of Networks and Systems* (MTNS' 2002), Notre Dame, 2002.
62. B.N. Parlett, Reduction to tridiagonal form and minimal realizations, *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 2, pp. 567-593, 1992.
63. B.N. Parlett, D.R. Taylor, and Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, *Math. Comp.*, vol. 44, pp. 105-124, 1985.
64. M. Robbé and M. Sadkane, A convergence analysis of GMRES and FOM methods for Sylvester equations, *Numer. Alg.*, 2002.
65. A. Ruhe, Rational Krylov sequence methods for eigenvalue computation, *Lin. Alg. Appl.*, vol. 58, pp. 391-405, 1984.
66. A. Ruhe, Rational Krylov algorithms for nonsymmetric eigenvalue problems. II: matrix pairs, *Lin. Alg. Appl.*, vol. 197/198, pp. 283-295, 1994.
67. Y. Saad, Projection and deflation methods for partial pole assignment in linear state feedback, *IEEE Trans. Automat. Control*, vol. 33, pp. 290-297, 1988.
68. Y. Saad, Numerical solutions of large Lyapunov equations, *Signal Processing, Scattering, Operator Theory, and Numerical Methods*, Birkhauser, pp. 503-511, 1990, (M.A. Kaashoek, J. H. Van Schuppen and A. C. Ran, Editors)
69. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, John Wiley, New York, 1992.
70. Y. Saad, Analysis of some Krylov Subspace approximations to the matrix exponential operator, *SIAM J. Numer. Anal.*, vol. 29, pp. 209-228, 1992b.
71. Y. Saad, *Iterative methods for sparse linear systems*, PWS, Boston, MA, 1996.
72. Y. Saad and M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.*, vol. 7, pp. 856-869, 1986.
73. M. Sadkane, Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems, *Numer. Math.*, vol. 64, pp. 195-211, 1993.
74. D. Sarkissian, *Theory and Computations of Partial Eigenvalue and Eigenstructure Assignment Problems in Matrix Second-order and Distributed Parameter Systems*, Ph.D. Dissertation, Northern Illinois University, DeKalb, Illinois, 2001.

75. V. Simoncini, On the numerical solution of $AX - XB = C$, *BIT*, vol. 36, pp. 814-830, 1996.
76. D.C. Sorensen, Implicit application of polynomial filters in a k -step Arnoldi method, *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 357–385, 1992.
77. D.C. Sorensen and A.C. Antoulas, *Projection methods for balanced model reduction*, unpublished manuscript, 2001 (available from the webpage: <http://www.ece.rice.edu/aca>).
78. G.W. Stewart, *Matrix Algorithms*, Vol. II: Eigensystems, SIAM, Philadelphia, 2001.
79. T.-J. Su and R.R. Craig, Jr., Model reduction and control of flexible structures using Krylov vectors, *J. Guidance Control Dynamics*, vol. 14, pp. 260-267, 1991.
80. P. Van Dooren, *The Lanczos Algorithm and Padé Approximations*, Short Course, Benelux Meeting on Systems and Control, 1995.
81. P. Van Dooren, *Gramian Based Model Reduction of Large-scale Dynamical Systems*, Short Course SIAM Annual Meeting, San Juan, Puerto Rico, July, 2000.
82. P. Van Dooren, Numerical linear algebra techniques for large scale matrix problems in systems and control, *Proc. IEEE Conf. Dec. Control*, 1992.
83. C. de Villemagne and R.E. Skelton, Model reductions using a projection formulation, *Int. J. Control*, vol. 46, pp. 2141-2169, 1987.
84. H.F. Walker, Implementation of the GMRES method using Householder Transformations, *SIAM J. Sci. Comput.*, vol. 9, pp. 152-163, 1988.
85. J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, UK, 1965.

Chapter 16

NUMERICAL METHODS FOR MATRIX-SECOND-ORDER CONTROL SYSTEMS

Contents

16.1 Introduction	896
16.2 First-order Representations	899
16.3 The Quadratic Eigenvalue Problem (QEP)	901
16.3.1 Linearization of the Quadratic Eigenvalue Problem	903
16.3.2 Solving the Quadratic Eigenvalue Problem.	905
16.3.3 Computation of the Partial Spectrum of the Quadratic Eigenvalue Problem: Shift and Invert Strategy and the Jacobi-Davidson Method	906
16.4 Independent Modal Control (IMSC) Approach	909
16.4.1 Modal Solution of the State Feedback Problem	911
16.4.2 Modal Solution of the Output Feedback Problem	911
16.4.3 Engineering and Computational Difficulties with the IMSC Approach	912
16.4.4 Simultaneous Diagonalization of the Pencil $K - \lambda M$.	913
16.4.5 Simultaneous Triangularization Under Modal Damping	914
16.5 Controllability, Observability, and Stability of Second-order Systems	915
16.5.1 Introduction	915
16.5.2 Eigenvalue Criteria of Controllability and Stabilizability	916
16.5.3 Stability of Second-order systems	918
16.6 Eigenvalue Bounds, Orthogonality of the Eigenvectors, and the Rayleigh Quotient of Quadratic Matrix Pencil	925
16.6.1 Eigenvalue Bounds for the Quadratic Pencil	925
16.6.2 Orthogonality of the Eigenvectors of the Quadratic Pencil	927

16.6.3 Rayleigh-Quotient Expressions for Quadratic Pencil	930
16.7 Nonmodal and Partial Modal Approaches to Stabilization, Partial Eigenvalue and Eigenstructure Assignments Using Feedback Control	933
16.7.1 Introduction	933
16.7.2 Problem Statements	934
16.7.3 A Nonmodal Solution of the Feedback Stabilization Problem	937
16.7.4 A Direct and Partial Modal Approach for Partial Eigenvalue and Eigenstructure Assignment	939
16.7.5 Robust Eigenvalue Assignment in Second-order System	952
16.8 Summary and Review	962
16.9 Chapter Notes and Further Reading	967

Topics Covered

- The Quadratic Eigenvalue Problem
- Controllability and Stability of Matrix Second-order Systems
- Eigenvalue Bounds and Orthogonality of Eigenvectors of Quadratic Matrix Pencil
- Nonmodal Algorithm for Stabilization in Matrix Secon-order System
- Direct and Partial-Modal Algorithms for Partial Eigenvalue and Eigenstructure Assignment in Matrix Second-order System
- Robust Eigenvalue Assignment in Matrix Second-order System

16.1 Introduction

The natural mathematical models of vibrations of strings, beams, plates and those of structures such as buildings, bridges, highways, air and space crafts are distributed parameter systems; that is, there are systems with distributed mass, stiffness, and damping parameters. The distributed parameter systems are also referred to as continuous or elastic or flexible systems. Because of lack of effective numerical methods for solving control problems directly in a distributed parameter setting, very often, in practice, a distributed parameter system is discretized to a system of matrix second-order differential equations of the form

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = f(t), \quad (16.1.1)$$

where M , D , and K are $n \times n$ real matrices and $f(t)$ is a vector.

In vibration applications, the matrices M , K , and D are known, respectively, as the **mass**, **stiffness**, and **damping** matrices. The vector $f(t)$ represents applied force or external excitation. The mass matrix is also known as the **inertia matrix**, because it arises from the inertial forces in the system. The damping and stiffness matrices, arise, respectively, from dissipative forces proportional to the velocity, and the elastic forces proportional to the displacement. If the system does not rotate and the motion is measured from the equilibrium position, then in most of the applications, the mass, stiffness, and damping matrices are symmetric; and furthermore, M is positive definite. Symmetric positive definite, positive semidefinite, negative definite and negative semidefinite matrices will be denoted respectively by $> 0, \geq 0, < 0, \leq 0$. Thus

$$M = M^T = \text{mass or inertia matrix } (M > 0)$$

$$K = K^T = \text{stiffness matrix}$$

$$D = D^T = \text{damping matrix}.$$

Further assumptions on the stiffness and damping matrices will be made, as the situations demand.

If the damping matrix D is absent, the system is called an **undamped** system; otherwise, it is called a **damped** system.

Controlling vibrations, especially dangerous vibrations such as resonance, is a fundamental problem in vibration engineering. This amounts to change the system responses, and, a natural way to do so is to apply a suitable control force to the structure. For example, by application of feedback control, system parameters such as stiffness and damping can be modified so that the closed-loop system has favorable responses to the external excitation. The feedback control can also help reduce the level of excitation transmitted to the structure (Soong (1990)).

If a control force vector $u(t)$ is applied to the system (16.1.1) with B as the control matrix, we then have the second-order control system:

$$M\ddot{x}(t) + D\dot{x}(t) + Kx(t) = f(t) + Bu(t).$$

An obvious approach to solve a control problem for the second-order control system is to construct a first-order representation of the second-order system, and then apply one of the standard well-established first-order techniques for that problem discussed in the previous chapters.

We will discuss the first-order representations of a second-order control system in the next section. As we will see there, the standard first-order representation requires the inversion of the mass matrix M ; thus, if this matrix is ill-conditioned, its inverse, and therefore, the state matrix of the first-order system, will not be computed correctly.

A non-standard first-order representation does not require the inversion of M . But, it gives rise to a descriptor control system (**see Chapter 5**) of the type

$$E\dot{x} = Ax + \hat{B}u.$$

Numerical methods for control problems of descriptor-control systems are not well-developed; especially, when the matrix E is ill-conditioned.

Furthermore, none of the first-order representations respect the exploitable properties such as definiteness, sparsity, etc., of the data matrices M , K , and D .

Another approach, widely known in the engineering literature, is the **Independent Modal Space Control** (IMSC) approach (see Inman (1989), Meirovitch (1990)).

As the name suggests, the IMSC approach aims at decomposing the problem into n **independent problems** by using the **modes** (eigenvectors) of the associated **quadratic eigenvalue problem**:

$$P(\lambda)x = (\lambda^2M + \lambda D + K)x = 0.$$

This eigenvalue problem, being quadratic, is difficult to solve in a numerically effective way. Indeed, numerically viable methods for computing the complete spectrum and the corresponding eigenvectors of the quadratic eigenvalue problem $P(\lambda)x = 0$, especially when the matrices M , K and D are large and sparse, are virtually nonexistent. The state-of-the-art techniques such as the Jacobi-Davidson method for the quadratic pencil or Lanczos and Arnoldi methods (applied to some suitable linear formulations of the quadratic eigenvalue problem) are capable of computing only a small number of eigenvalues and eigenvectors. See Bai et al. (2001) for details.

Our above discussions lead us to state that numerically viable algorithms for control problems of second-order systems ideally should be such that they (i) are capable of exploiting the properties of the coefficient matrices such as the sparsity, definiteness, etc., and (ii) can be implemented with only a minimal knowledge of the eigenvalues and eigenvectors (**frequencies** and **modes**) of the associated quadratic eigenvalue problem (possibly only with a small number of eigenvalues and eigenvectors that can be computed using the existing techniques or can be measured in a vibration laboratory). Some work to this effect on feedback control has been done recently.

In the last few years, nonmodal and partial-modal methods (methods that do not either require knowledge of eigenvalues and eigenvectors at all or only require a partial knowledge) have been developed for feedback stabilization (Datta and Rincón (1993)), partial pole placement problems (Datta, Elhay, and Ram (1997), Datta and Sarkissian (2001)), partial eigenstructure assignment problem (Datta, Elhay, Ram and Sarkissian (2000)), and robust pole placement problems (Chu and Datta (1996)). We will describe some of these recent work here.

Besides controlling vibrations, analysis of stability is also of fundamental importance. If the system is not stable, some suitable control force is usually applied to stabilize it. The stability of a system is determined by the nature of the eigenvalues, the explicit knowledge of the eigenvalues is not needed. The stability problem is classical.

There are some remarkable results on stability of a second-order system. Indeed, there are some authoritative books in this area (Lancaster (1966), Gohberg, Lancaster and Rodman (1982, 1983), etc.). We will briefly review some of these classical results and state and prove some new results in this Chapter. In particular, we will show information on the eigenvalues of the quadratic pencil $P(\lambda)$, such as **stability**, **eigenvalue bounds**, etc. can be extracted from partial knowledge of the eigenvalues of the matrices M , K , and D .

For notational convenience, we will sometimes denote $q(t)$ by q , $x(t)$ by x , $\dot{q}(t)$ by \dot{q} , $\ddot{q}(t)$ by \ddot{q} , etc.

The organization of this Chapter is as follows:

Section 16.2 contains first-order representations (both standard and non-standard) of the second-order system.

The associated quadratic eigenvalue problem is discussed in **Section 16.3**. The engineering and computational difficulties for using the Independent Modal Space Control (IMSC) approach are illustrated by means of state and output feedback problems in **Section 16.4**.

The concepts and basic results on controllability, stability and stabilizability are described in **Section 16.5**

Some theoretical bounds of eigenvalues and new recent results on the orthogonality of the eigenvectors and the Rayleigh quotient are contained in **Section 16.6**.

Section 16.7 is the main section of this chapter. Here the recent nonmodal and partial modal solutions of feedback stabilization, partial eigenvalue and partial eigenstructure assignments of a second-order system are described in details.

16.2 First-order Representations

In this section we show how the control system

$$M\ddot{q} + D\dot{q} + Kq = Bu \quad (16.2.1)$$

can be written in the first-order form

$$\dot{x} = Ax + \hat{B}u$$

and then discuss the numerical disadvantages of using this form in computational setting.

Define

$$x_1 = q \text{ and } x_2 = \dot{q}.$$

Then the above second-order system can be written as:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ M\dot{x}_2 &= -Dx_2 - Kx_1 + Bu. \end{aligned}$$

Let $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. Then we have

$$\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix} x + \begin{pmatrix} 0 \\ M^{-1}B \end{pmatrix} u \quad (16.2.2)$$

or

$$\dot{x} = Ax + \hat{B}u$$

where

$$\begin{aligned} A &= \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix}, \\ \hat{B} &= \begin{pmatrix} 0 \\ M^{-1}B \end{pmatrix}. \end{aligned} \tag{16.2.3}$$

The above representation is called the **standard first-order representation** of the second-order control system (16.2.1).

Numerical Difficulties with Standard First-order Representation

As is obvious from above that these are two principal numerical difficulties:

- 1 To compute the matrix A , the matrix M needs to be inverted. **If M is ill-conditioned, its inverse, and therefore the matrix A , will not be computed accurately.**
- 2 The important exploitable properties of the coefficient matrices M, K and D such as **definiteness** and **sparsity** (which are very often assets for large problems) are **completely destroyed in this formulation.**

Remarks: Although the standard first-order representation is not recommended to be used in solving a control problem associated with a second-order control system, this representation will be used throughout the rest of this chapter in the context of defining the basic concepts of **controllability, observability, stability, stabilizability**, etc. of a second-order system, and sometimes theoretical results will be established using this representation. Thus, **it is an important theoretical tool.**

Generalized First-order Representations

We now look into some other first-order representations that do not require the explicit inversion of M .

Let H be a nonzero arbitrary real matrix of order n . Then multiplying (16.2.2) on the left by the $2n \times 2n$ matrix

$$\begin{pmatrix} H & 0 \\ 0 & M \end{pmatrix}$$

and replacing x by z , it is easily seen that the equation (16.2.1) is transformed to:

$$\begin{pmatrix} H & 0 \\ 0 & M \end{pmatrix} \dot{z} = \begin{pmatrix} 0 & H \\ -K & -D \end{pmatrix} z + \begin{pmatrix} 0 \\ B \end{pmatrix} u \tag{16.2.4}$$

For different choices of the matrix H , we will then have different first-order representations.

For example, when $H = I$, we have

$$\begin{pmatrix} I & 0 \\ 0 & M \end{pmatrix} \dot{z} = \begin{pmatrix} 0 & I \\ -K & -D \end{pmatrix} z + \begin{pmatrix} 0 \\ B \end{pmatrix} u \tag{16.2.5}$$

When $H = M$, we have

$$\begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} \dot{z} = \begin{pmatrix} 0 & M \\ -K & -D \end{pmatrix} z + \begin{pmatrix} 0 \\ B \end{pmatrix} u \quad (16.2.6)$$

These are all nonsymmetric first-order representations. We can also have **symmetric** first-order representations. Take $H = -K$, then we have

$$\begin{pmatrix} -K & 0 \\ 0 & M \end{pmatrix} \dot{z} = \begin{pmatrix} 0 & -K \\ -K & -D \end{pmatrix} z + \begin{pmatrix} 0 \\ B \end{pmatrix} u \quad (16.2.7)$$

Numerical Difficulties with Non-standard First-order Representations

The non-standard first-order representations (16.2.4)–(16.2.7) are of the form:

$$E\dot{z} = Az + \hat{B}u, \quad (16.2.8)$$

which are known as **descriptor** systems (**See Chapter 5**).

Most of the numerical methods developed so far have been for the first-order systems of the form

$$\dot{x} = Ax + \hat{B}u.$$

Unfortunately, numerical methods for descriptor systems are not well established yet.

If the matrix E is nonsingular, then one can, of course, convert a descriptor system to a standard system by multiplying by E^{-1} on both sides; *however E can be ill-conditioned too and that will again result in inaccuracy.*

16.3 The Quadratic Eigenvalue Problem (QEP)

The solution of the system of second-order differential equations

$$M\ddot{q} + D\dot{q} + Kq = 0$$

gives rise to the following quadratic eigenvalue problem:

$$P(\lambda)x = (\lambda^2 M + \lambda D + K)x = 0.$$

This can be seen as follows.

Let $q(t) = xe^{\lambda t}$ be a solution of the second-order system, where x is a nonzero constant vector. Then the equation

$$M\ddot{q} + D\dot{q} + Kq = 0$$

becomes

$$(\lambda^2 M + \lambda D + K)xe^{\lambda t} = 0$$

or

$$(\lambda^2 M + \lambda D + K)x = 0.$$

Definition 16.3.1 The matrix $P(\lambda) = \lambda^2M + \lambda D + K$ is called a **quadratic pencil** or a **Lambda-matrix** (Lancaster (1966)).

Definition 16.3.2 A scalar $\lambda \in \mathbb{C}$ such that $\det(P(\lambda)) = 0$ is called an eigenvalue of the quadratic pencil $P(\lambda)$. The set of all eigenvalues is called the spectrum of $P(\lambda)$.

Note: There are $2n$ eigenvalues of the pencil $P(\lambda)$.

Definition 16.3.3 The non zero vectors x and y are, respectively, called the right and left eigenvectors, corresponding to the eigenvalue λ of the quadratic pencil $P(\lambda) = \lambda^2M + \lambda D + K$ if

$$(\lambda^2M + \lambda D + K)x = 0 \quad (16.3.1)$$

and

$$y^*(\lambda^2M + \lambda D + K) = 0, \quad (16.3.2)$$

where y^* is the conjugate transpose of the vector y .

Definition 16.3.4 The triplet (λ, x, y) is called the eigenpair of $P(\lambda)$.

Definition 16.3.5 The pairs (λ, x) and (λ, y) are called, respectively, right and left eigenpairs of P .

The *quadratic eigenvalue problem* is the problem of determining all the eigenvalues and the corresponding eigenvectors of the given quadratic pencil $P(\lambda)$. Note that the standard eigenvalue problem

$$Ax = \lambda x$$

and the generalized eigenvalue problem

$$Ax = \lambda Bx$$

are special cases of this problem (see (16.3.3) and (16.3.12)).

Definition 16.3.6 The pencil P is called singular if for any $\lambda \in \mathbb{C}$ the matrix $P(\lambda)$ is singular. Otherwise the pencil is called regular.

In this book we restrict ourselves to regular quadratic pencils.

16.3.1 Linearization of the Quadratic Eigenvalue Problem

The quadratic eigenvalue problem in the previous section can be solved by linearizing the nonlinear problem in two ways: transforming the problem to

- (i) a standard linear nonsymmetric eigenvalue problem, and
- (ii) a generalized eigenvalue problem.

Theorem 16.3.1 (Relation Between the Quadratic and the Standard Eigenvalue Problem).

A scalar $\lambda \in \mathbb{C}$ is an eigenvalue of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ with the corresponding right eigenvector x and the left eigenvector y if and only if λ is an eigenvalue of the matrix

$$A = \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix} \quad (16.3.3)$$

with the corresponding right eigenvector \hat{x} and the left eigenvector \hat{y} given by

$$\hat{x} = \begin{pmatrix} x \\ \lambda x \end{pmatrix} \text{ and } \hat{y} = \begin{pmatrix} \lambda M^* y + D^* y \\ M^* y \end{pmatrix}. \quad (16.3.4)$$

Proof: Since (λ, x, y) is an eigenpair of the quadratic pencil $P(\lambda)$, we must have

$$(\lambda^2 M + \lambda D + K)x = 0 \text{ and } y^*(\lambda^2 M + \lambda D + K) = 0. \quad (16.3.5)$$

From (16.3.5), it then follows that

$$A\hat{x} = \begin{pmatrix} \lambda x \\ -M^{-1}(K + \lambda D)x \end{pmatrix} = \lambda \begin{pmatrix} x \\ \lambda x \end{pmatrix} = \lambda \hat{x}$$

and

$$\hat{y}^* A = (-y^* K, \lambda y^* M + y^* D - y^* D) = (\lambda^2 y^* M + \lambda y^* D, \lambda y^* M) = \lambda \hat{y}^*,$$

which proves that $(\lambda, \hat{x}, \hat{y})$ is an eigenpair of the matrix A .

Next, suppose that λ is an eigenvalue of A and \hat{x} is the associated right eigenvector. Then

$$A\hat{x} = \lambda \hat{x}, \text{ where } \hat{x} = \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix}. \quad (16.3.6)$$

The equation (16.3.6) can be written as

$$\hat{x}_2 = \lambda \hat{x}_1 \quad (16.3.7)$$

and

$$-M^{-1}K\hat{x}_1 - M^{-1}D\hat{x}_2 = \lambda\hat{x}_2. \quad (16.3.8)$$

Substituting the equation (16.3.7) into the (16.3.8) and multiplying by M on the left, we get

$$-K\hat{x}_1 - \lambda D\hat{x}_1 = \lambda^2 M\hat{x}_1.$$

This shows that λ is the eigenvalue of P with the right eigenvector \hat{x}_1 .

Similarly, if \hat{y} is the right eigenvector of A associated with the eigenvalue λ , then

$$\hat{y}^* A = \lambda \hat{y}^*, \text{ where } \hat{y}^* = (\hat{y}_1^*, \hat{y}_2^*). \quad (16.3.9)$$

The equation (16.3.9) can be written as

$$-\hat{y}_2^* M^{-1} K = \lambda \hat{y}_1^* \quad (16.3.10)$$

and

$$\hat{y}_1^* - \hat{y}_2^* M^{-1} D = \lambda \hat{y}_2^*. \quad (16.3.11)$$

Substituting the equation (16.3.10) into the equation (16.3.11) after multiplication by λ we obtain

$$-(\hat{y}_2^* M^{-1}) K - \lambda(\hat{y}_2^* M^{-1}) D = \lambda^2 (\hat{y}_2^* M^{-1}) M,$$

which shows that $(\hat{y}_2^* M^{-1})$ is the left eigenvector. ■

Theorem 16.3.1 says that the $2n$ eigenvalues of the pencil $P(\lambda)$ are the eigenvalues of the matrix $A = \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix}$. Furthermore, if $\hat{x} = \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix}$ and $\hat{y} = (\hat{y}_1^*, \hat{y}_2^*)^*$ are, respectively, the right and left eigenvectors of A corresponding to an eigenvalue λ , then the respective right and left eigenvectors x and y of P are determined by

$$x = \hat{x}_1 \text{ and } y^* = \hat{y}_2^* M^{-1}.$$

Definition 16.3.7 *The eigenvalue λ of the matrix A is called semi-simple if it has the same algebraic and geometric multiplicities, that is, if it has m linearly independent eigenvectors, where m is multiplicity of the root λ of the characteristic polynomial $\det(A - sI)$ of the matrix A .*

Definition 16.3.8 *The eigenvalue λ of the quadratic pencil P is called semi-simple if λ is the semi-simple eigenvalue of the matrix A corresponding to this quadratic pencil.*

Requiring all eigenvalues to be semi-simple means that there is no need to introduce *Jordan chains* (also called the *associated vectors*) to describe the eigenstructure of our system. Since arbitrarily small perturbations can destroy the Jordan form, and for the reasons of numerical stability, **we restrict ourselves to eigenvalue problems with semi-simple eigenvalues**. The next theorem shows that the eigenvalue problem for the pencil $P(\lambda)$ is equivalent to the following generalized eigenvalue problem:

Theorem 16.3.2 (Relation Between the Quadratic and the Generalized Eigenvalue Problems).

A scalar $\lambda \in \mathbb{C}$ is an eigenvalue of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ with the corresponding right eigenvector x and the left eigenvector y if and only if λ is an eigenvalue of the linear pencil $A - \lambda B$, where

$$A = \begin{pmatrix} 0 & -K \\ -K & -D \end{pmatrix} \text{ and } B = \begin{pmatrix} -K & 0 \\ 0 & M \end{pmatrix}, \quad (16.3.12)$$

with $\hat{x} = \begin{pmatrix} x \\ \lambda x \end{pmatrix}$ and $\hat{y} = \begin{pmatrix} y \\ \lambda y \end{pmatrix}$ as right and left eigenvectors, respectively, corresponding to λ .

Proof: Proof of this theorem is analogous to the proof of Theorem 16.3.1.

Other Types of Generalized Eigenvalue Problems

The generalized eigenvalue problems given by Theorem 16.3.2 corresponds to (16.2.7). Other types of generalized eigenvalue problems are also possible.

Thus, corresponding to (16.2.5), the quadratic eigenvalue problem for the pencil $P(\lambda)$ is equivalent to the generalized eigenvalue problem for the linear pencil $A - \lambda B$, where

$$A = \begin{pmatrix} 0 & I \\ -K & -D \end{pmatrix}, \quad B = \begin{pmatrix} I & 0 \\ 0 & M \end{pmatrix}, \quad (16.3.13)$$

with $\hat{x} = \begin{pmatrix} x \\ \lambda x \end{pmatrix}$ and $\hat{y} = \begin{pmatrix} (\lambda M + D)^* y \\ y \end{pmatrix}$ as eigenvectors.

16.3.2 Solving the Quadratic Eigenvalue Problem.

Clearly, the quadratic eigenvalue problem for the pencil $P(\lambda)$ can be solved by solving the standard eigenvalue problem for the matrix A given by (16.3.3) or the generalized eigenvalue problems for the pair (A, B) given by (16.3.12) and (16.3.13).

Unfortunately, if the standard eigenvalue problem given by Theorem 16.3.1 is used to compute the eigenvalues and eigenvectors of the quadratic pencil $P(\lambda)$, then the matrix M has to be inverted, and, if it is ill-conditioned, then the eigenvalues and eigenvectors will not be computed accurately. Furthermore, all the exploitable properties such as the definiteness, sparsity, bandedness, etc., of the coefficient matrices M , D , and K , usually offered by a practical problem, will be completely destroyed.

Thus, the choice is between the generalized eigenvalue problems given by (16.3.12) and (16.3.13). If M , K and D are symmetric, the generalized eigenvalue problem defined by (16.3.12) is symmetric. However, the one given by (16.3.13) is not; although the matrix B is symmetric and symmetric positive definite if M is symmetric positive definite. The QZ algorithm described in

Chapter 4 can be used to solve these problems for dense matrices M , D , and K . The symmetric problem given by (16.3.12) will, however, be treated as a nonsymmetric problem, since the QZ algorithm is not designed to take advantage of the symmetry of the coefficient matrices.

The formulation (16.3.13) is used by the software package MSC/NASTRAN (Komzsik (1998)). As shown in **Theorem 16.3.2** the eigenvectors of the QEP can be recovered from the first or the last n components of those of the generalized eigenvalue problem. In finite precision arithmetic, these choices may not be equivalent. One may make this choice by finding whichever part yields the smallest backward error (See Tisseur (2000)).

For large and sparse problems, iterative methods such as the Lanczos or the Arnoldi methods (see **Chapter 15** for description of these basic algorithms) can be used. There are now several practical versions of these iterative methods. These include **Adaptively Lanczos Method** (ABLE), **Band Lanczos Method**, **Implicitly Restarted Arnoldi Method** (IRAM), block **Arnoldi Method**, etc. For details of these methods and the associated software see Bai, et al. (2000). *These methods are capable of computing only a few eigenvalues of the pencil $A - \lambda B$.* When M , K , and D are all symmetric, the symmetric indefinite Lanczos method (Parlett and Chen (1990)) is especially attractive for the formulation (16.3.12).

The implementation of the symmetric block Lanczos method by Grimes, Lewis and Simon (1994) is widely used.

An alternative to using the general eigenvalue methods is the **Jacobi-Davidson method** for the quadratic pencil to be described in Subsection 16.3.3.

An up-to-date account of the QEP can be found in Tisseur and Meerbergen (2001).

16.3.3 Computation of the Partial Spectrum of the Quadratic Eigenvalue Problem: Shift and Invert Strategy and the Jacobi-Davidson Method

We briefly review some methods that are commonly used to compute a part of the spectrum of the quadratic matrix pencil $P(\lambda)$ and discuss their advantages and limitations. The discussion has been taken from Bai et al. (2000).

Shifted and Invert Quadratic Eigenvalue Problem

The well known and popular “*shift and invert method*” is an iterative technique to compute a small number of eigenvalues and the corresponding eigenvectors of a matrix. For practical applications, it is useful that the structures of the matrices M , D , and K of the pencil $P(\lambda) = \lambda^2 M + \lambda D + K$, such as symmetry, bandedness, sparsity, etc., are exploited in computations. The shift and invert method is capable of doing so. In this subsection we discuss shift and invert strategy for the pencil $P(\lambda)$.

Using the shift $\lambda = \mu + \sigma$, the eigenvalue problem for the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ is transformed into the equivalent eigenvalue problem for the pencil $P_1(\mu)$ given by

$$P_1(\mu) = \mu^2 M + \mu(D + 2\sigma M) + (K + \sigma D + \sigma^2 M). \quad (16.3.14)$$

In particular, μ is an eigenvalue of $P_1(\mu)$ if and only if $\mu + \sigma$ is an eigenvalue of $P(\lambda)$, since $P(\mu + \sigma) = P_1(\mu)$.

Similarly, if 0 is not an eigenvalue of $P(\lambda)$, then the inversion $\lambda = 1/\mu$ transforms the eigenvalue problem for $P(\lambda)$ to the eigenvalue problem for the quadratic pencil

$$P_2(\mu) = \mu^2 K + \mu D + M. \quad (16.3.15)$$

This invert strategy is, thus, useful to compute some of the smallest (in modulus) eigenvalues and eigenvectors.

Combining the above shift and invert transformations (16.3.14) and (16.3.15), the quadratic eigenvalue problem for $P(\lambda)$ is transformed to the eigenvalue problem for

$$P_\sigma(\mu) = \mu^2(K + \sigma D + \sigma^2 M) + \mu(D + 2\sigma M) + M, \quad (16.3.16)$$

with $\mu = \frac{1}{\lambda - \sigma}$.

By definition, the pencil $P(\lambda)$ is regular if there exists a shift σ such that the matrix $P(\sigma) = K + \sigma D + \sigma^2 M$ is nonsingular.

Thus, the exterior eigenvalues μ of the quadratic pencil (16.3.15) will approximate the eigenvalues λ of the original pencil $P(\lambda)$, closest to the shift σ . The eigenvalues λ are then recovered from μ as

$$\lambda = \sigma + \frac{1}{\mu}.$$

The quadratic eigenvalue problems (16.3.14)-(16.3.16) can be solved by using their linearized formulations. For example, the generalized symmetric eigenvalue problem corresponding to (16.3.16) is

$$\begin{pmatrix} \hat{D} & \hat{K} \\ \hat{K} & 0 \end{pmatrix} \begin{pmatrix} x \\ (\lambda - \sigma)x \end{pmatrix} = \frac{1}{\lambda - \sigma} \begin{pmatrix} -\hat{M} & 0 \\ 0 & \hat{K} \end{pmatrix} \begin{pmatrix} x \\ (\lambda - \sigma)x \end{pmatrix}, \quad (16.3.17)$$

where $\hat{M} = \sigma^2 M + \sigma D + K$, $\hat{D} = D + 2\sigma M$ and $\hat{K} = M$.

If the matrices M , K , and D are real, but the shift σ is complex, it is better to use the transformation $(\lambda - \sigma)^{-1}(\lambda - \bar{\sigma})^{-1}$ (Parlett and Saad (1987)). In this case, most of the Lanczos or Arnoldi process can be carried out in real arithmetic.

The Jacobi-Davidson Method

Originally, the Jacobi-Davidson method was developed to compute the partial spectrum of the standard large and sparse eigenvalue problem

$$Ax = \lambda x. \quad (16.3.18)$$

It is based on the idea of projection. Given an orthogonal basis V of the low-dimensional search subspace, the large eigenvalue problem (16.3.18) is approximated by the “projected” eigenvalue problem

$$V^* A V s = \theta V^* V s. \quad (16.3.19)$$

The system (16.3.19) is a smaller eigenvalue problem and, therefore, can be efficiently solved by any of the standard methods. If (θ, s) is the eigenpair of (16.3.19), then the Ritz value θ is an approximate eigenvalue of (16.3.18) corresponding to the eigenvector approximated by the Ritz vector $u = Vs$.

The method is iterative in nature. At each iteration the basis V is expanded by the approximate solution $t \perp u$, of the correction equation

$$(I - uu^*)(A - \theta I)(I - uu^*)t = -r, \quad (16.3.20)$$

where $u = Vs$ and $r = Au - \theta u$. For stability reasons, the basis of the search subspace is constructed to be orthonormal. The new basis vector is the orthogonal complement of t with respect to the previous basis vectors.

It can be shown that if the correction equation (16.3.20) is solved with sufficient accuracy, then the asymptotic rate of convergence to the eigenpair of (16.3.18) is at least quadratic. In practice, however, it is often more efficient to approximate the solution to the equation (16.3.20) by a fast iterative numerical method, such as, for example, a small number of GMRES steps (see **Chapter 15**). Although this increases the number of Jacobi-Davidson iterations, each iteration becomes considerably less expensive. **If the desired eigenvalue is well separated from the other eigenvalues, then the Jacobi-Davidson method converges almost quadratically.** In other cases, the convergence is linear and its rate depends on the relative separation of the desired eigenvalue.

The idea can be generalized to compute the partial spectrum of the quadratic pencil $P(\lambda)$. Thus, the large quadratic eigenvalue problem for the pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ is projected onto a low-dimensional subspace spanned by the columns of V , which leads to the small eigenvalue problem for the projected quadratic pencil

$$P_V(\theta) = \theta^2 V^* M V + \theta V^* D V + V^* K V \quad (16.3.21)$$

that can be solved by any direct method. The best (for example, closest to some target value or with the largest real part) eigenvalue θ of this projected eigenproblem is selected. If s is the right eigenvector corresponding to θ , then the Ritz vector $u = Vs$ approximates the eigenvector of $P(\lambda)$ corresponding to the approximate eigenvalue θ of $P(\lambda)$.

If the approximation is not satisfactory, then the search subspace spanned by the orthogonal columns of the matrix V has to be expanded by a vector t that is determined by the Jacobi-Davidson correction equation

$$\left(I - \frac{pu^*}{u^*p} \right) P(\theta)(I - uu^*)t = -r \quad (16.3.22)$$

where $u = Vs$, $r = P(\theta)u = (\theta^2 M + \theta D + K)u$, and $p = P'(\theta)u = (2\theta M + D)u$. Then V is replaced by the result of the modified Gram-Schmidt method applied to the matrix (V, t) .

The process is repeated until the desired eigenpair is detected, that is until the residual vector r becomes small enough. If the dimension of the search subspace becomes too large, then the

process could be restarted with a new search subspace determined by the few best eigenpairs of the projected low-dimensional problem (16.3.21).

Based on the above discussion we can state the following algorithm:

Algorithm 16.3.1 Jacobi-Davidson Method for the Quadratic Eigenvalue Problem.

Inputs: The $n \times n$ matrices M, C , and K and tolerance ε .

Outputs: An eigenvalue θ and right eigenvector x of the quadratic matrix pencil $P(\lambda) = \lambda^2 M + \lambda D + K$.

Step 1. Choose an $n \times m$ orthonormal matrix V , and compute $W_0 = KV$, $W_1 = DV$, $W_2 = MV$ and $M_i = V^*W_i$ for $i = 0, 1, 2$.

Step 2. Iterate steps 3 through 9 until convergence.

Step 3. Compute the right eigenpairs (θ, s) of the projected pencil

$$(\theta^2 M_2 + \theta M_1 + M_0)s = 0.$$

Step 4. Select the desired eigenpair (θ, s) with $\|s\|_2 = 1$.

Step 5. Compute $u = Vs$, $r = P(\theta)u$ and $p = P'(\theta)u$.

Step 6. If $\|r\| < \varepsilon$ then $\lambda = \theta, x = u$, STOP.

Step 7. Solve (approximately) the following linear system for $t \perp u$:

$$\left(I - \frac{pu^*}{u^*p} \right) P(\theta)(I - uu^*)t = -r.$$

Step 8. Orthogonalize t against V , $v = t/\|t\|_2$ and compute for $i = 0, 1, 2$:

$$w_0 = Kv, \quad w_1 = Dv, \quad w_2 = Mv \text{ and } M_i = \begin{pmatrix} M_i & V^*w_i \\ v^*W_i & v^*w_i \end{pmatrix}.$$

Step 9. Expand $V = (V, v)$ and $W_i = (W_i, w_i)$, $i = 0, 1, 2$.

16.4 Independent Modal Control (IMSC) Approach

As the title suggests, the basic idea behind the Independent Modal Space Control (IMSC) approach is to **decouple** the control system into n **independent systems** using the eigenvectors (**modes**) of the associated quadratic pencil.

Suppose that $M = M^T > 0$, $K = K^T \geq 0$, we then know (see Datta (1995)), that there exists a matrix S_m such that

$$\begin{aligned} S_m^T M S_m &= I \\ S_m^T K S_m &= \Lambda_k = \text{diag } (\omega_1^2, \dots, \omega_n^2), \end{aligned} \tag{16.4.1}$$

where $\omega_i^2, i = 1, \dots, n$ are the eigenvalues of the matrix $K - \lambda M$.

The numbers $\omega_i, i = 1, \dots, n$ are called the **natural frequencies**. The process of finding the matrix S_m is called the technique of **simultaneous diagonalization**.

An algorithm for simultaneous diagonalization of M and K , is given at the end of this section. The equations (16.4.1) show that a symmetric definite linear pencil $K - \lambda M$ can be decoupled. To decouple a damped second-order system we need that the damping matrix D also be diagonalized by the same transforming matrix S_m , as well.

Assuming that D is also symmetric, this will happen if and only if the following commutativity condition is satisfied (Inman (1989)):

$$DM^{-1}K = KM^{-1}D \quad (16.4.2)$$

The damping structure satisfying this relation is called **modal damping**. A particular case of modal damping is **Rayleigh damping**, which is defined as

$$D = \alpha M + \beta K, \quad (16.4.3)$$

where α , and β are any real scalars. Assuming **modal damping**, that is, the damping matrix D is such that the above commutativity condition is satisfied, we will have

$$S_m^T D S_m = \Lambda_d = \text{diag}(d_1, \dots, d_n) = \text{diag}(2\zeta_1\omega_1, 2\zeta_2\omega_2, \dots, 2\zeta_n\omega_n). \quad (16.4.4)$$

The quantities $\zeta_i = \frac{d_i}{2\omega_i}, i = 1, \dots, n$, are called the **modal damping ratios**. The matrix S_m is called the **modal transforming matrix**.

Define now the new vector $z(t)$ by

$$q(t) = S_m z(t).$$

Then premultiplying by S_m^T the system

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = Bu(t) = f(t)$$

is transformed to the following **modal control system**:

$$I\ddot{z}(t) + \Lambda_d \dot{z}(t) + \Lambda_k z(t) = S_m^T Bu = \tilde{B}u \quad (16.4.5)$$

and if we define $u = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix}$, and $S_m^T B = (\tilde{b}_{ij})$, $z = \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix}$,

then

$$\ddot{z}_i(t) + 2\zeta_i\omega_i\dot{z}_i(t) + \omega_i^2 z_i(t) = \tilde{f}_i(t), \quad i = 1, \dots, n, \quad (16.4.6)$$

where

$$\tilde{f}_i(t) = \tilde{b}_{i1}u_1 + \tilde{b}_{i2}u_2 + \tilde{b}_{i3}u_3 + \tilde{b}_{i4}u_4 + \dots + \tilde{b}_{in}u_n. \quad (16.4.7)$$

These equations have the “**appearance**” of being decoupled, *but in reality they are not*, because of the form of the right hand side. For example, if $u = Fz$, as in the case of state feedback control (to be discussed later in the Chapter), the problem is coupled. However, if $\tilde{f}_i(t)$ are not inter-related, then the above is a set of n **decoupled equations**, which can be used to solve a given control problem in a “**simple-to-solve**” way.

We illustrate the *practical and computational difficulties of the IMSC approach* to solve control problems by means of the state-feedback and output-feedback eigenvalue assignment problems, assuming that solutions to these feedback problems exist.

16.4.1 Modal Solution of the State Feedback Problem

Let's first consider the state-feedback problem.

Given the modal system:

$$I\ddot{y} + \Lambda_d\dot{y} + \Lambda_k y = S_m^T B u,$$

if we choose $u = F_1\dot{y} + F_2y$, then the **modal closed-loop system** becomes:

$$I\ddot{y} + (\Lambda_d - S_m^T B F_1)\dot{y} + (\Lambda_k - S_m^T B F_2)y = 0.$$

Thus, the closed-loop system will decouple for individual eigenvalue-assignment if and only if the matrices $S_m^T B F_1$ and $S_m^T B F_2$ are both diagonal.

In case this happens, the modal closed-loop system reduces to n decoupled-equations:

$$\ddot{z}_i + (2\zeta_i\omega_i - \alpha_i)\dot{z}_i + (\omega_i^2 - \beta_i)z_i = 0, i = 1, 2, \dots, n,$$

where α_i and β_i are the diagonal entries of the matrices $S_m^T B F_1$ and $S_m^T B F_2$, respectively.

These n independent equations can now be solved for α_i , and β_i , given a desired set of eigenvalues to be assigned. Once α_i and β_i are known, the feedback matrices F_1 and F_2 can be obtained from the equations:

$$\begin{aligned} S_m^T B F_1 &= \text{diag } (\alpha_1, \dots, \alpha_n) \\ S_m^T B F_2 &= \text{diag } (\beta_1, \dots, \beta_n). \end{aligned}$$

16.4.2 Modal Solution of the Output Feedback Problem

We next consider the **output feedback problem**.

Consider the modal system with the output equation:

$$I\ddot{z} + \Lambda_d\dot{z} + \Lambda_k z = S_m^T B u$$

$$y = C_p S_m z + C_\vartheta S_m \dot{z}$$

If u is chosen as $u = -Gy$, then the modal closed-loop system is

$$I\ddot{z} + (\Lambda_d + S_m^T BK_\vartheta S_m)\dot{z} + (\Lambda_K + S_m^T BK_p S_m)z = 0$$

where

$$K_p = GC_p, \quad K_\vartheta = GC_\vartheta.$$

The modal closed-loop system will decouple if and only if the matrices $S_m^T BK_\vartheta S_m$ and $S_m^T BK_p S_m$ are both diagonal matrices.

If this happens, then we will have n decoupled-equations:

$$\ddot{z}_i + (2\zeta_i \omega_i + \alpha'_i)\dot{z}_i + (\omega_i^2 + \beta'_i)z_i = 0, i = 1, \dots, n.$$

where

$$S_m^T BK_\vartheta S_m = \text{diag } (\alpha'_1, \dots, \alpha'_n),$$

and

$$S_m^T BK_p S_m = \text{diag } (\beta'_1, \dots, \beta'_n).$$

These last equations can be solved for α'_i and β'_i , given a desired set of eigenvalues or some modal response information.

Once α'_i and β'_i are known, the above equations can be solved for the output feedback matrices K_ϑ and K_p .

16.4.3 Engineering and Computational Difficulties with the IMSC Approach

We have illustrated the IMSC approach of control problems with the state and output feedback problems. Other control problems can similarly be solved. For discussions of the use of IMSC in solutions of other control problems, see Inman (1989) (see also **Exercises 1, 3, and 4**). As seen from the above discussions, the IMSC approach has some serious engineering and computational difficulties.

First, it relies on simultaneous diagonalization of the matrix-triple M , K , and D . As we have seen before, for this to happen, the commutativity relation

$$DM^{-1}K = KM^{-1}D$$

must be satisfied (see (16.4.2)).

For general damped second-order systems, this is unrealistic.

The Rayleigh damping ($D = \alpha M + \beta K$) is unpractical and is used mostly for computational convenience.

Second, the decoupling of the closed-loop systems requires simultaneous diagonalization of certain matrix expressions involving the control and feedback matrices. For example, in the

output-feedback case, both the matrices $S_m^T BK_\vartheta S_m$ and $S_m^T BK_p S_m$ have to be diagonal. In case the matrices BK_ϑ and BK_p both happen to be symmetric, this will be true if and only if

$$BK_\vartheta M^{-1} D = DM^{-1} BK_\vartheta$$

and

$$BK_p M^{-1} K = KM^{-1} BK_p.$$

“Unfortunately, this puts very stringent requirements on the location and number of sensors and actuators” (Inman (1989), pp. 173).

Third, there are some serious **numerical difficulties** in the process of the simultaneous diagonalization technique itself, as stated below.

16.4.4 Simultaneous Diagonalization of the Pencil $K - \lambda M$.

To understand these numerical difficulties, we state below the algorithm for simultaneous diagonalization of the pair (M, K) . For a proof of the algorithm, see Datta (1995, pp. 510).

Algorithm 16.4.1 Simultaneous Diagonalization of the Symmetric Definite Pencil $K - \lambda M$

Inputs: $M, K \in \mathbb{R}^{n \times n}$.

Outputs: $S_m \in \mathbb{R}^{n \times n}$ such that S_m is nonsingular, $S_m^T M S_m = I$ and $S_m^T K S_m$ is diagonal.

Assumption: $M = M^T > 0$, $K = K^T \geq 0$.

Step 1. Compute the Cholesky factorization of M :

$$M = LL^T.$$

Step 2. Use the QR algorithm to diagonalize $C = L^{-1}KL^{-T}$:

$$C = Q \text{diag } (\omega_1^2, \dots, \omega_n^2) Q^T$$

Step 3. Compute $S_m = L^{-T}Q$.

Computational Difficulties with Simultaneous Diagonalization Technique

1. Steps 1 and 2 are **computationally prohibitive** if M and K are large and sparse.

2. Clearly S_m is not orthogonal, in general. The computed ω_i^2 's can be inaccurate when M is ill-conditioned. It can be shown (Golub and Van Loan (1996)) that

$$\omega_i^2 \in \lambda(L^{-1}KL^{-T} + E_i)$$

where $\|E_i\|_2 \cong \mu\|K\|_2\|M^{-1}\|_2$ (μ is the machine precision).

This implies that diagonalization is not recommended when $\|M^{-1}\|_2$ is too large.

3. Even when M and K are sparse, the modal transforming matrix S_m is dense, in general.

16.4.5 Simultaneous Triangularization Under Modal Damping

The primary advantage of modal damping is that it makes it possible to transform the matrices M, K , and D simultaneously to diagonal forms.

Unfortunately, as said above, the transforming matrix S_m , in general, is not orthogonal and can be highly ill-conditioned.

One, therefore, wonders if it is possible to transform the triple (M, K, D) to any other suitable condensed forms using orthogonal transforming matrices.

The following result, proved by Williams and Laub (1992), shows that *under the assumption of modal damping*, the triple (M, K, D) can be transformed to upper triangular matrices using orthogonal transformations; however, *this is not possible under general damping* [Exercise 5].

Theorem 16.4.1 (Simultaneous Triangularization using Orthogonal Transformations)

Let $M = M^T > 0$ and $K = K^T \geq 0$. Then under the assumption of modal damping, there exist orthogonal matrices Q and Z such that

$$Q^T M Z = T_m$$

$$Q^T D Z = T_d$$

$$\text{and } Q^T K Z = T_k,$$

where T_m, T_d , and T_k are upper triangular matrices.

Proof: Consider the following decompositions of the inverse modal transforming matrix, S_m^{-1} :

$$S_m^{-T} = QR \text{ (QR decomposition)}$$

and

$$S_m^{-T} = ZL \text{ (Z orthogonal and L lower triangular)}$$

Then

$$M = S_m^{-T} S_m^{-1} = Q R L^T Z^T$$

or

$$Q^T M Z = R L^T = T_m, \text{ upper triangular.}$$

Similarly, $Q^T D Z = R \operatorname{diag}(2\zeta_1\omega_1, 2\zeta_2\omega_2, \dots, 2\zeta_n\omega_n) L^T = T_d$, upper triangular and $Q^T K Z = R \operatorname{diag}(\omega_1^2, \dots, \omega_n^2) L^T = T_k$, upper triangular.

Remarks: It has been proved in Williams and Laub (1992) that the *modal damping is also necessary for simultaneous orthogonal triangularization* if all the frequencies ω_i are distinct.

Algorithm 16.4.2 Simultaneous Triangularization of the Modal Pencil $\lambda^2 M + \lambda D + K$.

Inputs: The matrices M, K, D , each of order n .

Outputs: Orthogonal transforming matrices Q and Z ; and the upper triangular matrices T_m, T_d , and T_k , such that $Q^T M Z = T_m$, $Q^T D Z = T_d$, and $Q^T K Z = T_k$.

Assumptions

1. $M = M^T > 0$, $K = K^T \geq 0$, $D = D^T \geq 0$.
2. D is a **modal damping matrix**, that is, D satisfies: $DM^{-1}K = KM^{-1}D$.

Step 1. Compute the modal transforming matrix S_m using Algorithm 16.4.1.

Step 2. Compute the QR and QL factorizations of the matrix S_m^{-T} :

$$\begin{aligned} S_m^{-T} &= QR \text{ (QR factorization),} \\ S_m^{-T} &= ZL \text{ (QL factorization).} \end{aligned}$$

(Q and Z are orthogonal, and R , and L are, respectively upper and lower triangular matrices).

Step 3. Compute

$$\begin{aligned} T_m &= Q^T M Z \\ T_d &= Q^T D Z \\ T_k &= Q^T K Z. \end{aligned}$$

16.5 Controllability, Observability, and Stability of Second-order Systems

16.5.1 Introduction

The concepts of controllability, observability, stability, stabilizability, and detectability, etc. of a second-order system are defined in terms of the respective concepts of an associated first-order representation. Here we present some criteria that do not, however, require any transformation to a first-order form.

16.5.2 Eigenvalue Criteria of Controllability and Stabilizability

The following criteria of controllability and stabilizability are due to Laub and Arnold (1984). The criteria can be used to test controllability and stabilizability with respect to each eigenvalue individually. Other conditions of controllability and stabilizability were derived earlier by Hughes and Skelton (1980) [Exercises 1].

Theorem 16.5.1 *The system*

$$M\ddot{q} + D\dot{q} + Kq = Bu \quad (16.5.1)$$

is controllable if and only if

$$\text{rank } (P(\lambda), B) = \text{rank}(\lambda^2 M + \lambda D + K, B) = n \text{ for all } \lambda,$$

where λ is an eigenvalue of the quadratic pencil $P(\lambda)$.

The above system is **stabilizable** if and only if $\text{rank}(\lambda^2 M + \lambda D + K, B) = n$ for all eigenvalues λ with nonnegative real parts of the pencil $P(\lambda)$.

Proof: Consider the following generalized first-order representation of the second-order model (corresponding to $H = I$ in (16.2.4)):

$$\begin{pmatrix} I & 0 \\ 0 & M \end{pmatrix} \dot{z}(t) = \begin{pmatrix} 0 & I \\ -K & -D \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ B \end{pmatrix} u(t)$$

or

$$E\dot{z}(t) = Az(t) + \hat{B}u(t).$$

Now, this descriptor first-order system is controllable if and only if $2n = \text{rank}(A - \lambda E, \hat{B})$, for all λ which are the generalized eigenvalues of the pencil $A - \lambda E$. Again,

$$\begin{aligned} 2n &= \text{rank}(A - \lambda E, \hat{B}) \\ &= \text{rank} \begin{pmatrix} -\lambda I & I & 0 \\ -K & -D - \lambda M & B \end{pmatrix} \\ &= \text{rank} \begin{pmatrix} \lambda M + D & I \\ I & 0 \end{pmatrix} \begin{pmatrix} -\lambda I & I & 0 \\ -K & -D - \lambda M & B \end{pmatrix} \begin{pmatrix} -I & 0 & 0 \\ -\lambda I & I & 0 \\ 0 & 0 & I \end{pmatrix} \\ &= \text{rank} \begin{pmatrix} \lambda^2 M + \lambda D + K & 0 & B \\ 0 & I & 0 \end{pmatrix} \end{aligned}$$

Clearly this is true if and only if

$$\text{rank}(\lambda^2 M + \lambda D + K, B) = n.$$

The stabilizability criterion can similarly be established. ■

Remarks: Note that the above criteria allow one to test for controllability and stabilizability for selected eigenvalues of the system. **Also, note that the controllability implies stabilizability, but not conversely.**

Controllability of the pair $(P(\lambda), B)$. As in the first-order system, the controllability of the second-order system (16.5.1) will be referred to as the controllability of the pair $(P(\lambda), B)$.

Example 16.5.1 Consider the undamped second-order system:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \ddot{q} + \begin{pmatrix} 1 & 2 & 3 \\ 2 & 10 & 4 \\ 3 & 4 & 15 \end{pmatrix} q = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} u.$$

Then

$$M = \text{diag}(1, 2, 3),$$

$$K = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 10 & 4 \\ 3 & 4 & 15 \end{pmatrix}, \quad \text{and } B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

The eigenvalues of the pencil

$$\begin{pmatrix} 0 & I \\ -K & 0 \end{pmatrix} - \lambda \begin{pmatrix} I & 0 \\ 0 & M \end{pmatrix}$$

are:

$$\lambda_1 = -2.7216j, \quad \lambda_2 = 2.7216j, \quad \lambda_3 = -1.8385j, \quad \lambda_4 = 1.8385j, \quad \lambda_5 = 0.4616j, \quad \lambda_6 = -0.4616j$$

$$\text{rank}(\lambda_i^2 M + K, B) = 3, \quad i = 1, 2, \dots, 6.$$

Thus we conclude that the system is **controllable** and, therefore, **stabilizable**.

Partial Controllability

Since each eigenvalue of the system can be checked individually for controllability, it makes sense to talk about controllability of the pair $(P(\lambda), B)$ with respect to a given eigenvalue or a given set of eigenvalues.

Thus, we define partial controllability as:

Corollary 16.5.1 The pair $(P(\lambda), B)$ is partially controllable with respect to the eigenvalue set Ω if and only if $\text{rank}(P(\lambda), B) = n$ for all $\lambda \in \Omega$.

16.5.3 Stability of Second-order systems

A Review of Basic Concepts of Stability

The concepts of stability, asymptotic stability, bounded-input, bounded-output (BIBO) stability, etc., discussed in Chapter 7, can be easily carried over to the second-order system.

Perhaps, it is more convenient to define these concepts in terms of the corresponding concepts for the standard first-order representation of the second-order system.

Recall that the standard first-order representation of the unforced damped second-order system:

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = 0 \quad (16.5.2)$$

is given by

$$\dot{x}(t) = Ax(t),$$

where

$$A = \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix}, \quad x(t) = \begin{pmatrix} q(t) \\ \dot{q}(t) \end{pmatrix}.$$

Let $x(0)$ be the vector of initial conditions for the system.

The system has a **stable equilibrium** if for any arbitrary positive number ϵ , there exists a positive number δ , depending upon ϵ , such that whenever $\|x(0)\| < \delta$, $\|x(t)\| < \epsilon$ for all $t > 0$. **The physical significance of the above definition is that for a bounded displacement of the initial position, the motion of the system remains bounded for all time t .**

A stable equilibrium is said to be “**asymptotically stable**” if

$$\lim_{t \rightarrow \infty} \|x(t)\| = 0.$$

Note that asymptotic stability implies stability, but not conversely.

Bounded-Input, Bounded-Output Stability

As in the case of first-order system, the stability of the forced response of a system is defined in terms of the bounds of the response vector $x(t)$.

A forced system is **bounded stable** with respect to the given input $\tilde{f}(t)$, if $x(t)$ is bounded for any initial condition $x(0)$, that is, $\|x(t)\| < C$ where C is some positive integer.

The system is **bounded-input, bounded-output** (BIBO) stable if for any bounded $\tilde{f}(t)$, the corresponding $x(t)$ remains bounded.

Note that if the system is BIBO stable, then it is bounded stable; but not conversely.

The Eigenvalue Criteria of Stability

Since the eigenvalues of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ are those of the matrix A of the standard first order representation, the following eigenvalue criterion of the asymptotic stability of a second-order system can be immediately proved [**Exercise 5**].

Theorem 16.5.2 (Eigenvalue criterion for Asymptotic Stability) *The system is asymptotically stable if and only if all the eigenvalues of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ have negative real parts.*

In the following we present some **nonspectral** criteria of stability. Some of these criteria are classical and some are recent.

We will also derive some computable bounds for the eigenvalues of the pencil $P(\lambda)$.

But, before doing so, we derive some theoretical results on the eigenvalues of the pencil $P(\lambda)$, which will be used in derivation of nonspectral stability criteria and the eigenvalue bounds.

Expressions for Real and Imaginary Parts of the Eigenvalues

The following theorem establishes some theoretical results on the expressions for real and imaginary parts of the eigenvalues of a quadratic pencil. The results have been taken from Datta and Rincón (1993).

Theorem 16.5.3 *Let $\lambda = \alpha + i\beta$ be an eigenvalue of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$, and x be the corresponding eigenvector. Let T_x stand for $x^* T x$, where $x^* = (\bar{x})^T$.*

Assume that

$$M = M^T > 0, \quad K = K^T \geq 0 \quad \text{and} \quad D = D^T \geq 0.$$

Then

$$(i) \quad \alpha = \frac{-|\lambda|^2 D_x}{|\lambda|^2 M_x + K_x}$$

$$(ii) \quad \beta^2 = \frac{K_x}{M_x} - \left(\frac{D_x}{2M_x} \right)^2 \quad (\text{if } \beta \neq 0).$$

Proof of (i).

Since (λ, x) is an eigenpair of the pencil $P(\lambda) = \lambda^2 M + \lambda D + K$, we must have

$$(\lambda^2 M + \lambda D + K)x = 0.$$

Multiplying by x^* on the left gives

$$\lambda^2 M_x + \lambda D_x + K_x = 0.$$

Multiplying now by $\bar{\lambda} = (\alpha - i\beta)$, we obtain

$$(\alpha + i\beta)|\lambda|^2 M_x + |\lambda|^2 D_x + (\alpha - i\beta)K_x = 0$$

Equating the real and imaginary parts on both sides, gives

$$\alpha|\lambda|^2 M_x + |\lambda|^2 D_x + \alpha K_x = 0$$

$$\beta|\lambda|^2 M_x - \beta K_x = 0.$$

Solving for α , we immediately obtain (i).

Proof of (ii).

If $\beta \neq 0$, from the last expression, we have

$$|\lambda|^2 = \frac{K_x}{M_x}.$$

Since $\alpha^2 + \beta^2 = |\lambda|^2$, we then obtain

$$\alpha^2 + \beta^2 = \frac{K_x}{M_x}.$$

That is, $\beta^2 = \frac{K_x}{M_x} - \alpha^2$

Now by (i) $\alpha = \frac{-|\lambda|^2 D_x}{|\lambda|^2 M_x + K_x} = \frac{-\frac{K_x}{M_x} D_x}{2K_x} = -\frac{D_x}{2M_x}$ (using $|\lambda|^2 = \frac{K_x}{M_x}$).

Thus $\beta^2 = \frac{K_x}{M_x} - \frac{D_x^2}{4M_x^2} = \frac{K_x}{M_x} - \left(\frac{D_x}{2M_x}\right)^2$.

Note The assumptions that M is symmetric positive definite and K is positive semidefinite guarantee that the denominators in (i), (ii), do not become zero. ■

Nonspectral Criteria of Stability

There is a vast literature in this area. We just give here a very brief overview. We start with a very well-known classical stability-criterion called the Rayleigh criterion. The proof of this criterion is usually given using the concept of Lyapunov functions (see, e.g., Inman (1989), Chapter 4). We will give the proof here using the simple expressions for real and imaginary parts of the eigenvalues of $P(\lambda)$ that we derived in the last section.

Theorem 16.5.4 (Rayleigh Criterion of Stability)

Let M, K , and D be symmetric and positive definite matrices. Then the system (16.5.2) is asymptotically stable.

Proof: Let M_x, K_x , and D_x be the same as in Theorem 16.5.3. Let $\lambda = \alpha + i\beta$ be an eigenvalue of $P(\lambda) = \lambda^2 M + \lambda D + K$, then we have, from Theorem 16.5.3:

$$\alpha = \frac{-|\lambda|^2 D_x}{|\lambda|^2 M_x + K_x},$$

where x is the eigenvector corresponding to λ .

Clearly, since M, K , and D are symmetric and positive definite, M_x, D_x , and K_x are all positive. Thus α is negative, proving that the system (16.5.2) is stable. ■

Overdamping, Underdamping and Stability.

An interesting situation arises when the matrices M, K, D are symmetric and positive definite and the system is **overdamped**. Then not only the eigenvalues have negative real parts, but they are real as well.

The system (16.5.2) is **overdamped** if

$$D_x^2 > 4M_x K_x$$

for all nonzero x .

Analogously, the system (16.5.2) is defined to be **underdamped** if $D_x^2 < 4M_x K_x$, for all nonzero x .

The following result is due to Duffin (1960).

Theorem 16.5.5 *Let $M = M^T > 0, K = K^T > 0$, and $D = D^T > 0$. Then (i) if the system (16.5.2) is overdamped, all the eigenvalues of $P(\lambda)$ are real and negative. (ii) If the system is underdamped, all the eigenvalues of $P(\lambda)$ are nonreal and have negative real parts.*

Proof: Since M is symmetric positive definite, we can take $M = I$, without any loss of generality. The pencil then becomes

$$P(\lambda) = \lambda^2 I + \lambda D + K$$

Let λ_0 be an eigenvalue of $P(\lambda)$ and u be the corresponding unit eigenvector; that is $\det(P(\lambda_0)) = 0$, $\|u\| = 1$.

Then $u^* P(\lambda_0) u = \lambda_0^2 + \lambda_0(u^* D u) + u^* K u = 0 = \lambda_0^2 + \lambda_0 D_u + K_u = 0$,

which gives $\lambda_0 = \frac{-D_u \pm \sqrt{D_u^2 - 4K_u}}{2}$.

Overdamping then implies that λ_0^2 is real. Similarly, underdamping implies that λ_0 is nonreal. The negativity of the eigenvalues follows from the Rayleigh Criterion. (**Theorem 16.5.4**). ■

It is difficult to numerically verify the criterion of overdamping as stated above. The following sufficient condition of overdamping due to Barkwell and Lancaster (1992) is rather verifiable numerically. See also the recent work of Higham, Tisseur, and Van Dooren (2002) where they have derived numerical algorithms to check the overdamping property.

Theorem 16.5.6 Let M, K , and D be symmetric and positive definite. Let β_1 and ν_n be the minimal and maximal eigenvalues of $M^{-1}D$ and $M^{-1}K$, respectively. If

$$\beta_1 > 2(\nu_n)^{\frac{1}{2}}$$

then the system is overdamped.

Note: The above condition requires the knowledge of eigenvalues of $M^{-1}D$ and $M^{-1}K$. The following condition does not require any knowledge of eigenvalues (see Barkwell and Lancaster (1992)).

The system is overdamped if

$$D > kM + \frac{1}{k}K$$

for some $k > 0$.

Finally, we give another characterization of overdamping. This characterization is due to Veselić (1993).

Theorem 16.5.7 Let M, K , and D be symmetric and positive definite.

Define

$$N = \begin{pmatrix} 0 & L_2^T L_1^{-1} & 0 \\ L_1^{-1} L_2 & L_1^{-1} & D L_1^{-T} \end{pmatrix},$$

where L_1 and L_2 are the Cholesky factors of K and M ; that is,

$$M = L_2 L_2^T, \quad K = L_1 L_1^T.$$

Then the system is overdamped if and only if there exists a real number μ such that the matrix $N - \mu J$ is positive definite, where

$$J = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix}.$$

We next consider the undamped system.

Theorem 16.5.8 Let $M = M^T > 0, K = K^T \geq 0$, and $D = 0$. Then the system is **marginally stable**; that is, all the eigenvalues of the pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ are purely imaginary.

Proof: The proof is immediate from the expression of α in Theorem 16.5.3. Note that when $D = 0, \alpha = 0$. ■

Note: If both M and K are positive definite, but $D = 0$, then it can be shown (Lancaster and Tismenetsky (1985)) that not only the eigenvalues of $P(\lambda)$ are all purely imaginary, but also have linear elementary divisors.

Positive Semidefinite Damping and Stability

The last theorem tells us that an **undamped system cannot be asymptotically stable**. On the other hand, from the Rayleigh criterion of stability, we know that if damping matrix D is symmetric and positive definite, then the system is asymptotically stable.

The interesting question, therefore, arises is to what happens if D is merely positive semidefinite. In the following we first state a spectral criterion and then state and prove a nonspectral criterion, in case the damping matrix D is positive semidefinite.

Theorem 16.5.9 (Spectral Criterion of Asymptotic Stability with Positive Semidefinite Damping). *Let $M = M^T > 0, D = D^T \geq 0, K = K^T > 0$, then the system (16.5.2) is asymptotically stable if, and only if,*

$$\text{rank} (\lambda^2 M + K, D) = n,$$

for all $\lambda \in \mathbb{C}$.

Proof: Wimmer (1974). ■

The above criterion is a spectral one; the following is a non-spectral characterization.

Theorem 16.5.10 (Nonspectral Criterion of Asymptotic Stability with Positive Semidefinite Damping) *The system (16.5.2) with $M = M^T > 0, K = K^T > 0$, and $D = D^T \geq 0$ is asymptotically stable if, and only if, the pair*

$$(KM^{-1}, D)$$

is controllable.

Proof: From Theorem 16.5.9, we have

$$\text{rank}(\lambda^2 M + K, D) = n$$

for all $\lambda \in \mathbb{C}$. Since M is nonsingular, the above condition is equivalent to

$$\text{rank}(\lambda^2 I + M^{-1}K, M^{-1}D) = n$$

for all $\lambda \in \mathbb{C}$. This means that

$$(M^{-1}K, M^{-1}D)$$

is controllable. Again, $(M^{-1}K, M^{-1}D)$ is controllable if, and only if,

$$(KM^{-1}, D)$$

is controllable. ■

The above result was originally proved by Walker and Schmitendorf (1973). See also Wimmer (1974).

Oscillations of the Second-order Systems

We have seen in previous sections that a great deal of qualitative information, such as stability, can be extracted without computing the eigenvalues explicitly. In this section, we will see that several important results on oscillations of a vibratory system can also be obtained without explicit knowledge of the eigenvalues.

Consider the damped system:

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = 0$$

where $M = M^T > 0$, $D = D^T > 0$, and $K = K^T > 0$.

In this case, we know that the system is asymptotically stable.

However, how the response of such systems oscillates depends upon the nature of damping, that is, if the system is **critically damped**, **overdamped**, or **underdamped**. We have defined before the overdamped and underdamped systems.

Analogously, we define a **critically damped system** as the one for which $D_x^2 = 4M_xK_x$.

The following results on oscillations then can be proved (see Inman (1989), pp. 62).

Oscillations of An Asymptotically Stable System

Theorem 16.5.11 *Consider the system:*

$$M\ddot{q}(t) + D\dot{q}(t) + Kq = 0,$$

with symmetric positive definite matrices M , D , and K .

- (a) *If the system is **critically damped**, then each eigenvalue is a repeated negative real number, and the eigenvectors are real. The response of the system does not oscillate.*
- (b) *If the system is **overdamped**, then each eigenvalue is a negative real number, and the eigenvectors are real. The response of the system does not oscillate.*
- (c) *If the system is **underdamped**, each eigenvalue is a complex number with negative real part (occurring in complex conjugate pairs) and the eigenvectors are, in general, complex (unless the system is modal, that is, unless $DM^{-1}K = KM^{-1}D$; in which case the eigenvectors are all real). The response of the system oscillates with decaying amplitude.*

16.6 Eigenvalue Bounds, Orthogonality of the Eigenvectors, and the Rayleigh Quotient of Quadratic Matrix Pencil

In this section, we derive some recent results on bounds of the eigenvalues, the orthogonality of the eigenvectors, and the Rayleigh-quotient-like expressions for the quadratic eigenvalue pencil $P(\lambda) = \lambda^2 M + \lambda D + K$.

The eigenvalue bounds are computable, even for large and sparse systems, in the sense that these bounds are obtained from the spectral radii of the coefficient matrices M , K , and D , which are symmetric, and as mentioned before, there are numerically effective methods for computing these quantities of symmetric matrices, even for very large and sparse problems. *The results on orthogonality of the eigenvectors will be used later to obtain partial-modal solution of the partial pole-placement and eigenstructure assignment problems, and in fact, one of these results plays there a key role.*

The Rayleigh-quotient results are of independent interests and important in their own rights. The orthogonality results of the eigenvectors and the Rayleigh-quotient results are generalizations of some of the corresponding results for linear pencil $K - \lambda M$.

16.6.1 Eigenvalue Bounds for the Quadratic Pencil

Here we derive two **computable** bounds for the eigenvalues of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$. Our assumptions are:

$$\begin{aligned} M &= M^T > 0, \quad K = K^T \geq 0, \\ D &= D^T \geq 0. \end{aligned}$$

Other computable bounds for the eigenvalues of matrix polynomials have been recently derived by Higham and Tisseur (2001a).

The quantities $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$, respectively, represent the largest and the smallest eigenvalue of a symmetric matrix A , and $\rho(A)$ is the spectral radius of A .

We first state two lemmas that are well-known in the linear algebra literature. They are related to the Rayleigh-quotients. Proof of the first Lemma can be found in Parlett (1980), and that of the second one in Stewart and Sun (1990).

Lemma 16.6.1 *Let $A \in \mathbb{R}^{n \times n}$, $A = A^T$ and $x \in \mathbb{C}^n$ with $\|x\|_2 = 1$. Then*

$$\lambda_{\min}(A) \leq x^* Ax \leq \lambda_{\max}(A).$$

Lemma 16.6.2 *Let $A \in \mathbb{R}^{n \times n}$, A normal (that is $A^T A = A A^T$), and $x \in \mathbb{C}^n$ with $\|x\|_2 = 1$. Then*

$$|x^* Ax| \leq \rho(A),$$

where $\rho(A)$ is the spectral radius of A .

Theorem 16.6.1 below is proved using Lemmas 16.6.1 and 16.6.2.

Theorem 16.6.1 If λ is an eigenvalue of $P(\lambda) = \lambda^2 M + \lambda D + K$, then

$$|\lambda| \leq \frac{\rho(D)}{2\lambda_{\min}(M)} + \sqrt{\left(\frac{\rho(D)}{2\lambda_{\min}(M)}\right)^2 + \frac{\lambda_{\max}(K)}{\lambda_{\min}(M)}}.$$

Proof: Let (λ, x) be an eigenpair for $P(\lambda)$. Then

$$P(\lambda)x = 0.$$

Premultiplying by x^* , we have

$$(x^* M x) \lambda^2 = -(x^* D x) \lambda - x^* K x.$$

Taking norms of both sides of and applying the triangle inequality (see Chapter 1), we get

$$(x^* M x) |\lambda|^2 \leq (|x^* D x|) |\lambda| + x^* K x$$

which can be rewritten as

$$(x^* M x) |\lambda|^2 - (|x^* D x|) |\lambda| - x^* K x \leq 0.$$

Solving the inequality for $|\lambda|$, we obtain

$$|\lambda| \leq \frac{|x^* D x|}{2(x^* M x)} + \sqrt{\left(\frac{|x^* D x|}{2(x^* M x)}\right)^2 + \frac{x^* K x}{x^* M x}}.$$

The proof of the Theorem now follows immediately from the above Lemmas. ■

Remarks: Note that the bound in Theorem 16.6.1 can be obtained by knowing the smallest and largest eigenvalues of M, K , and D . As said before, these quantities can be computed in rather numerically effective way even for large and sparse matrices.

Theorem 16.6.2 If λ is an eigenvalue of $P(\lambda)$, then

$$|\lambda| \leq \frac{1}{2}(\|M^{-1}D\|_2) + \sqrt{\left[\frac{1}{2}(\|M^{-1}D\|_2)\right]^2 + \|M^{-1}K\|_2}.$$

Proof: Let (λ, x) be an eigenpair for $P(\lambda)$, with $\|x\|_2 = 1$. Then

$$P(\lambda)x = \mathbf{0}.$$

Premultiplying by M^{-1} , we have

$$\lambda^2 x = -\lambda M^{-1} D x - M^{-1} K x.$$

Now, taking norms on both sides and applying the triangle inequality again, we get

$$|\lambda|^2 \|x\|_2 \leq |\lambda| \|M^{-1}Dx\|_2 + \|M^{-1}Kx\|_2.$$

Since $\|x\|_2 = 1$, we have

$$|\lambda|^2 \leq |\lambda| \|M^{-1}D\|_2 + \|M^{-1}K\|_2.$$

Thus

$$|\lambda|^2 - \|M^{-1}D\|_2 |\lambda| - \|M^{-1}K\|_2 \leq 0.$$

Solving now this inequality for $|\lambda|$, we obtain the result. ■

16.6.2 Orthogonality of the Eigenvectors of the Quadratic Pencil

Before stating the results on the orthogonality of the eigenvectors of the quadratic pencil $P(\lambda)$, we first remind the readers the corresponding result for an ordinary symmetric matrix A and that for a symmetric-definite linear pencil $K - \lambda M$. (Recall that the pencil $K - \lambda M$ is symmetric-definite if $M = M^T > 0$ and $K = K^T \geq 0$).

Theorem 16.6.3 (Orthogonality of the Eigenvectors of Symmetric Matrix)

Let A be a symmetric matrix. Then its eigenvectors can be chosen to be orthonormal.

In other words, if A is symmetric, then there exists an orthogonal matrix Q such that $Q^T A Q = D$, where D is a diagonal matrix with the diagonal entries as the eigenvalues.

Proof: See Datta (1995, pp. 378). ■

Theorem 16.6.4 (Orthogonality of the eigenvectors of Symmetric-Definite Linear Pencil)

Let $K - \lambda M$ be an $n \times n$ symmetric positive definite pencil; that is, $M = M^T > 0$ and $K = K^T \geq 0$. Then the n eigenvalues $\mu_i, i = 1, \dots, n$ are all real.

Furthermore, the eigenvectors x_i can be chosen so that the following orthogonality relations hold:

$$\left. \begin{array}{l} x_i^T K x_j = 0 \\ x_i^T M x_j = 0 \end{array} \right\} i \neq j$$

and

$$x_i^T M x_i = 1, \quad x_i^T K x_i = \mu_i.$$

In other words, there exists an eigenvector matrix X such that

$$X^T M X = I, \quad \text{and } X^T K X = \text{diag}(\mu_1, \dots, \mu_n).$$

Proof: See Datta (1995, 506-507). ■

Note: The algorithm for simultaneous diagonalization of the matrix pair (M, K) given in section 16.4.4 is based on the above Theorem.

For the general quadratic pencil $P(\lambda)$, the above orthogonality relations hold only if the damping is modal, that is, if $KM^{-1}D$ is symmetric.

We now derive **three** orthogonality relations for the symmetric quadratic pencil $P(\lambda)$, without any assumption on modal damping. The results of Theorem 16.6.5 and Theorem 16.6.6 have been taken from Datta, Elhay and Ram (1997).

Theorem 16.6.5 (Orthogonality of the Eigenvectors of Quadratic Pencil)

Let $P(\lambda) = \lambda^2 M + \lambda D + K$, where $M = M^T > 0$, $D = D^T$, and $K = K^T$.

Let $X \in \mathbb{C}^{n \times 2n}$, and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_{2n})$ be, respectively, the eigenvector matrix and the eigenvalue matrix of the pencil $P(\lambda)$. **Assume that the eigenvalues $\lambda_1, \dots, \lambda_n$ are all distinct and different from zero.** Then there exist diagonal matrices D_1, D_2 , and D_3 such that

- (a) $\Lambda X^T M X \Lambda - X^T K X = D_1$
- (b) $\Lambda X^T D X \Lambda + \Lambda X^T K X + X^T K X \Lambda = D_2$
- (c) $\Lambda X^T M X + X^T M X \Lambda + X^T D X = D_3$.

Furthermore

- (d) $D_1 = D_3 \Lambda$
- (e) $D_2 = -D_1 \Lambda$
- (f) $D_2 = -D_3 \Lambda^2$.

Proof: By definition, the pair (X, Λ) must satisfy the $n \times 2n$ system of equations (the **eigendecomposition** of the pencil $P(\lambda) = \lambda^2 M + \lambda D + K$) :

$$MX\Lambda^2 + DX\Lambda + KX = 0.$$

Isolating the term in D , we have from above

$$-DX\Lambda = MX\Lambda^2 + KX.$$

Multiplying this on the left by ΛX^T gives

$$-\Lambda X^T DX\Lambda = \Lambda X^T MX\Lambda^2 + \Lambda X^T KX$$

Taking the transpose gives

$$-\Lambda X^T DX \Lambda = \Lambda^2 X^T MX \Lambda + X^T KX \Lambda$$

Now, subtracting the latter from the former gives, on rearrangement,

$$\Lambda X^T MX \Lambda^2 - X^T KX \Lambda = \Lambda^2 X^T MX \Lambda - \Lambda X^T KX$$

or

$$(\Lambda X^T MX \Lambda - X^T KX) \Lambda = \Lambda (\Lambda X^T MX \Lambda - X^T KX).$$

Thus, the matrix $\Lambda X^T MX \Lambda - X^T KX$ which we denote by D_1 , must be diagonal since it commutes with the diagonal matrix Λ , the diagonal entries of which are distinct. We thus have the first orthogonality relation (a):

$$\boxed{\Lambda X^T MX \Lambda - X^T KX = D_1}$$

Similarly, isolating the term in M of the eigendecomposition equation, we get

$$-MX\Lambda^2 = DX\Lambda + KX,$$

and multiplying this on the left by $\Lambda^2 X^T$ gives

$$-\Lambda^2 X^T MX \Lambda^2 = \Lambda^2 X^T DX \Lambda + \Lambda^2 X^T KX.$$

Taking the transpose, we have

$$-\Lambda^2 X^T MX \Lambda^2 = \Lambda X^T DX \Lambda^2 + X^T KX \Lambda^2.$$

Subtracting the last equation from the previous one and adding $\Lambda X^T KX \Lambda$ to both sides gives, after some rearrangement,

$$\Lambda(\Lambda X^T DX \Lambda + \Lambda X^T KX + X^T KX \Lambda) = (\Lambda X^T DX \Lambda + \Lambda X^T KX + X^T KX \Lambda) \Lambda$$

Again, this commutativity property implies, since Λ has distinct diagonal entries, that

$$\boxed{\Lambda X^T DX \Lambda + \Lambda X^T KX + X^T KX \Lambda = D_2}$$

is a diagonal matrix. This is the *second orthogonality relation (b)*.

The first and second orthogonality relations together easily imply the *third orthogonality relation (c)*:

$$\boxed{\Lambda X^T MX + X^T MX \Lambda + X^T DX = D_3}$$

To prove (d), we multiply the last equation on the right by Λ to obtain

$$\Lambda X^T MX \Lambda + X^T MX \Lambda^2 + X^T DX \Lambda = D_3 \Lambda,$$

which, using the eigendecomposition equation, gives

$$\Lambda X^T M X \Lambda + X^T (-K X) = D_3 \Lambda.$$

So, from the first orthogonality relation we see that

$$D_1 = D_3 \Lambda$$

Next, using the eigendecomposition equation, we rewrite the second orthogonality relations as

$$D_2 = \Lambda X^T (DX\Lambda + KX) + X^T KX\Lambda = \Lambda X^T (-MX\Lambda^2) + X^T KX\Lambda = (-\Lambda X^T MX\Lambda + X^T KX)\Lambda$$

By the first orthogonality relation we then have

$$D_2 = -D_1 \Lambda$$

Finally, from $D_1 = D_3 \Lambda$ and $D_2 = -D_1 \Lambda$

we have

$$D_2 = -D_3 \Lambda^2.$$

We remind the reader that matrix and vector transposition here does not mean conjugation for complex quantities. ■

Corollary 16.6.1 *Let $\Lambda = \text{diag}(\Lambda_1, \Lambda_2)$, where $\Lambda_1 = \text{diag}(\lambda_1, \dots, \lambda_p)$ and $\Lambda_2 = \text{diag}(\lambda_{p+1}, \dots, \lambda_{2n})$. Similarly, let $X = (X_1, X_2)$, where $X_1 = (x_1, x_2, \dots, x_p)$ and $X_2 = (x_{p+1}, \dots, x_{2n})$. Let the sets $\{\lambda_1, \dots, \lambda_p\}$ and $\{\lambda_{p+1}, \dots, \lambda_{2n}\}$ be disjoint.*

Then

$$\Lambda_1 X_1^T M X_2 \Lambda_2 - X_1^T K X_2 = 0$$

Proof: By Part (a) of Theorem 16.6.5, we have

$$(\Lambda X^T M X \Lambda - X^T K X) \Lambda = \Lambda (\Lambda X^T M X - X^T K X).$$

That is,

$$D_1 \Lambda = \Lambda D_1, \text{ where } D_1, \text{ is the matrix under the parenthesis.}$$

Let $D_1 = (d_{ij})$. Thus, from the last relation, we have $d_{ij} = 0$ if $1 \leq i \leq p \leq j \leq 2n$. In matrix notation, this implies the relation that is required to prove. ■

16.6.3 Rayleigh-Quotient Expressions for Quadratic Pencil

It is instructive to view the orthogonality relations component-wise, which lead to Rayleigh-quotient-like expressions. Before stating and proving these Rayleigh-quotient like expressions for the quadratic pencil, we remind the readers the corresponding result for the symmetric matrix and that for the symmetric definite pencil.

Rayleigh-Quotient for Symmetric Matrix

Let A be an $n \times n$ symmetric matrix and that x be a given nonzero n -vector. Then the quotient

$$\lambda = R_q(x) = \frac{x^T A x}{x^T x}$$

is called the **Rayleigh-quotient** of the vector x .

Theorem 16.6.6 (The Rayleigh-Quotient Theorem for Symmetric Matrix)

Let A be a symmetric matrix. Then, the pair (λ, x) as defined above, is such that $f(\lambda) = \|(A - \lambda I)x\|$ is minimized. In other words, if x is an approximate eigenvector, then $\lambda = R_q(x)$ is a reasonable choice for the corresponding eigenvalue.

Proof: See Datta ((1995), pp. 415). ■

The Rayleigh-Quotient for Symmetric-Definite Linear Pencil

We now state the Rayleigh-Quotient Theorem for a symmetric definite linear pencil.

Consider the $n \times n$ **Symmetric-definite linear pencil**: $K - \lambda M$. Let x be a nonzero n -vector. Then the quotient

$$\lambda = R_{gq}(x) = \frac{x^T K x}{x^T M x}$$

is called the **generalized Rayleigh-Quotient** of the vector x .

The following result on the generalized Rayleigh-quotient holds.

Theorem 16.6.7 (The Generalized Rayleigh-Quotient Theorem)

The Rayleigh-Quotient $\lambda = R_{gq}(x) = \frac{x^T K x}{x^T M x}$ minimizes $f(\lambda) = \|Kx - \lambda Mx\|_B$ where $\|\cdot\|_B$ is defined as $\|z\|_B^2 = z^T B^{-1} z$. In other words, if x is an approximate eigenvector of the pencil $Kx = \lambda Mx$, then $\lambda = R_{gq}(x)$ is a reasonable choice for the corresponding generalized eigenvalue.

Proof: See Golub and Van Loan ((1996), pp. 465). ■

Rayleigh-Quotient Expressions for Quadratic Pencil

We now derive a Rayleigh-Quotient like expression for quadratic pencil.

Theorem 16.6.8 (The Rayleigh-Quotient Theorem for Quadratic Pencil)

Let $(x_i, \lambda_i), i = 1, \dots, 2n$ be the eigenpairs of the symmetric definite quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$, and that $\lambda_1, \dots, \lambda_{2n}$ be all distinct.

Then provided the denominators do not vanish,

$$(a) \quad \left. \begin{array}{lcl} \lambda_i \lambda_j & = & \frac{x_i^T K x_j}{x_i^T M x_j} \\ -\frac{\lambda_i + \lambda_j}{\lambda_i \lambda_j} & = & \frac{x_i^T D x_j}{x_i^T K x_j} \\ -(\lambda_i + \lambda_j) & = & \frac{x_i^T D x_j}{x_i^T M x_j} \end{array} \right\}, \text{ for each } 1 \leq i \neq j \leq 2n.$$

and

$$(b) \quad \left. \begin{array}{lcl} \lambda_i & = & \frac{x_i^T (\lambda_i^2 M - K) x_i}{x_i^T (2\lambda_i M + D) x_i} \\ -\lambda_i & = & \frac{x_i^T (\lambda_i^2 D + 2\lambda_i K) x_i}{x_i^T (\lambda_i^2 M - K) x_i} \\ -\lambda_i^2 & = & \frac{x_i^T (\lambda_i^2 D + 2\lambda_i K) x_i}{x_i^T (2\lambda_i M + D) x_i} \end{array} \right\}, \text{ for each } i = 1, 2, \dots, 2n.$$

Proof: Equating the off-diagonal entries of the matrices on both sides of the three orthogonality relations, proved in Theorem 16.6.5, we have, for $i \neq j$;

$$\begin{aligned} x_i^T (\lambda_i \lambda_j M - K) x_j &= 0 \\ x_i^T (\lambda_i \lambda_j D + (\lambda_i + \lambda_j) K) x_j &= 0 \\ x_i^T ((\lambda_i + \lambda_j) M + D) x_j &= 0 \end{aligned}$$

The part (a) of the Theorem now immediately follows by rewriting these relations.

Similarly, the part (b) follows from the three relationship between D_1, D_2 , and D_3 in Theorem 16.6.5. ■

Derivation of the Rayleigh-Quotient for the Symmetric-Definite Linear Pencil from Theorem 16.6.8

If we let $D = 0$ in $P(\lambda) = \lambda^2 M + \lambda D + K$,

then we have the pencil $P(\mu) = K - \mu M$, where $\mu = -\lambda^2$.

The Rayleigh-quotient for this generalized symmetric definite pencil is then

$$\mu = R_{qq} = \frac{x^T K x}{x^T M x},$$

which is immediately recovered from the last equation of Part (b) in Theorem 16.6.8 by setting $D = 0$.

16.7 Nonmodal and Partial Modal Approaches to Stabilization, Partial Eigenvalue and Eigenstructure Assignments Using Feedback Control

16.7.1 Introduction

As we have seen in Section 16.2 and Section 16.4 that there are two standard approaches for solving a control problem modeled by a matrix second-order system:

First, via transformation to one of the first-order state-space forms (16.2.2)-(16.2.7), followed by application of a numerically effective technique, developed for the first-order state-space problem in the preceding chapters.

Second, using the Independent Modal Space Control (IMSC) technique discussed in Section 16.4.

The engineering and computational difficulties associated with each of the above two approaches have been discussed.

From these discussions, we conclude that a feedback control problem of a second-order system in practice should be solved (i) without its transformation to a first-order form and (ii) using either no knowledge or only a partial knowledge of eigenvalues and eigenvectors of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$.

In the context of (ii), note that, besides the others, one of the principal computational difficulties of the IMSC approach is that it requires complete knowledge of the spectrum and the eigenvectors of $P(\lambda)$; whereas *numerically viable methods for computing the whole spectrum of $P(\lambda)$, especially for large and sparse problems, have not yet been developed* (**Section 16.3**). As seen there the state-of-the-art techniques such as the symmetric indefinite Lanczos method and the Jacobi-Davidson method are capable of computing only a few exterior or selected eigenvalues of the quadratic pencil.

In view of these, attempts have been made in the last few years to develop feedback control algorithms for a matrix second-order system that *neither requires a transformation to a first-order system nor the knowledge of the complete spectrum and the associated eigenvectors of the pencil $P(\lambda)$* . Remarkable progress has been made on several feedback control problems: **direct and partial-modal approach** has been developed for feedback stabilization (**Algorithm 16.7.1**), for partial eigenvalue assignment (**Algorithm 16.7.2** and **Algorithm 16.7.3**) and partial eigenstructure assignment (**Algorithm 16.7.4**).

The approach is **direct** because each problem is solved in its own second-order form without resorting to a first-order transformation. Thus, the structures offered by practical applications such as sparsity, definiteness and bandness, etc., can be exploited in computation.

The approach is **partial-modal** in the sense that it requires only a partial knowledge of eigenvalues and eigenvectors of the associated quadratic matrix pencil. In view of the fact that a part of the spectrum and the corresponding eigenvectors can be computed using the state-of-the-art

techniques, even for large and sparse problems, the “*partial-modal*” nature makes the approach completely viable for practical applications.

In this section, we will briefly describe these methods.

In Subsection 16.7.2, we state the problems.

In Subsection 16.7.3, we describe a nonmodal approach for feedback stabilization.

In Subsection 16.7.4, we describe a partial-modal approach for the partial eigenvalue and eigenstructure assignment problems.

Finally, in Subsection 16.7.5, we describe a solution to the robust eigenstructure assignment problem, which is a generalization to the second-order model of the well-known Kautsky, Nichols and Van Dooren algorithm for the first-order problem described in Chapter 11.

The following notations will be used throughout the Subsections 16.7.2–16.7.4.

$\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_{2n})$ - matrix of the eigenvalues of the open-loop pencil $P(\lambda)$.

$\Lambda_1 = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_p)$ - matrix of the eigenvalues to be reassigned.

$\Lambda_2 = \text{diag}(\lambda_{p+1}, \lambda_{p+2}, \dots, \lambda_{2n})$ - matrix of the eigenvalues to remain invariant.

$\Lambda'_1 = \text{diag}(\mu_1, \dots, \mu_p)$ - matrix of the new eigenvalues.

$X = (x_1, x_2, \dots, x_{2n})$ - matrix of the eigenvectors of $P(\lambda)$.

$X_1 = (x_1, x_2, \dots, x_p)$ - matrix of eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_p$.

$X_2 = (x_{p+1}, x_{p+2}, \dots, x_{2n})$ - matrix of eigenvectors corresponding to the eigenvalues $\lambda_{p+1}, \lambda_{p+2}, \dots, \lambda_{2n}$.

$Y_1 = (y_1, y_2, \dots, y_p)$ - matrix of the new eigenvectors.

16.7.2 Problem Statements

Consider the system (16.2.1):

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = Bu(t), \quad (16.7.1)$$

where

$u(t)$ - input (control) vector,

B - $n \times m$ control matrix.

Suppose the vectors $\dot{q}(t)$ and $q(t)$ are known and we want to modify the behavior of the controlled system by choosing the input vector u appropriately.

If we choose $u = F^T\dot{q}(t) + G^Tq(t)$, then the closed-loop system becomes

$$M\ddot{q}(t) + (D - BF^T)\dot{q}(t) + (K - BG^T)q(t) = 0. \quad (16.7.2)$$

In order to stabilize the above system, the feedback matrices F^T and G^T should be chosen appropriately so that the resulting system is asymptotically stable.

Denote the closed-loop pencil associated with the above closed-loop system by

$$P_c(\lambda) = \lambda^2 M + \lambda(D - BF^T) + (K - BG^T). \quad (16.7.3)$$

Feedback Stabilization Problem for Second-order System

Given

1. Real $n \times n$ matrices $M = M^T > 0$, $D = D^T$, $K = K^T$.
2. The $n \times m (m \leq n)$ control matrix B .

Find real feedback matrices F and G such that the closed-loop pencil $P_c(\lambda)$ is asymptotically stable.

In a realistic situation, however, only a few eigenvalues are “troublesome”; so it makes more sense to alter only those “troublesome” eigenvalues, while keeping the rest of the spectrum invariant. This leads to the following problem, known as the *partial eigenvalue assignment problem* for the quadratic pencil $P(\lambda)$.

Partial Eigenvalue Assignment Problem For Second-order System

Given

1. Real $n \times n$ matrices $M = M^T > 0$, $D = D^T$, $K = K^T$.
2. The $n \times m (m \leq n)$ control matrix B .
3. The self-conjugate subset $\{\lambda_1, \dots, \lambda_p\}$, $p < n$ of the open-loop spectrum $\{\lambda_1, \dots, \lambda_p; \lambda_{p+1}, \dots, \lambda_{2n}\}$ and the corresponding eigenvector set $\{x_1, \dots, x_p\}$.
4. The self-conjugate set $\{\mu_1, \dots, \mu_p\}$ of numbers.

Find *real* feedback matrices F and G such that the spectrum of the closed-loop pencil $P_c(\lambda)$ is $\{\mu_1, \dots, \mu_p; \lambda_{p+1}, \dots, \lambda_{2n}\}$.

While the above problem is important in its own right, it is to be noted that, if the system response needs to be altered by feedback, both eigenvalue assignment as well as eigenvector assignment should be considered. *This is because the eigenvalues determine the rate at which system response decays or grows while the eigenvectors determine the shape of the response.* Recall that such a problem is called the *Eigenstructure Assignment Problem*. Unfortunately, a complete set of eigenvectors cannot, in general, be assigned, when B is given a priori (see, Chapter 10 and the paper by Andry, et al. (1986)). This consideration leads to the following more tractable feedback control problem for the quadratic pencil, known as the *partial eigenstructure assignment problem*.

Partial Eigenstructure Assignment Problem for Second-order System

Given

1. Real $n \times n$ matrices $M = M^T > 0$, $D = D^T$, $K = K^T$.
2. The self-conjugate subset $\{\lambda_1, \dots, \lambda_p\}$, $p < n$ of the open-loop spectrum $\{\lambda_1, \dots, \lambda_p; \lambda_{p+1}, \dots, \lambda_{2n}\}$ and the corresponding eigenvector set $\{x_1, \dots, x_p\}$.
3. The self-conjugate sets of numbers and vectors $\{\mu_1, \dots, \mu_p\}$ and $\{y_1, \dots, y_p\}$, such that $\mu_j = \overline{\mu_k}$ implies $y_j = \overline{y_k}$.

Find a *real* control matrix B of order $n \times m$ ($m \leq n$), and *real* feedback matrices F and G such that the spectrum of the closed-loop pencil is $\{\mu_1, \dots, \mu_p; \lambda_{p+1}, \dots, \lambda_{2n}\}$ and the eigenvector set is $\{y_1, \dots, y_p; x_{p+1}, \dots, x_{2n}\}$, where x_{p+1}, \dots, x_{2n} are the eigenvectors corresponding to $\lambda_{p+1}, \dots, \lambda_{2n}$.

Finally, we state the **robust eigenvalue assignment problem**. As we have seen before in Chapter 11, even when a numerically stable algorithm is used to solve an eigenvalue assignment problem, the eigenvalues of the **computed closed-loop** pencil may be very different from the eigenvalues that are needed to be assigned. Our analysis in Chapter 11 on the conditioning of the eigenvalue assignment problem shows that the *conditioning of the eigenvector matrix of the closed-loop matrix is a major contributing factor to this effect*.

Thus, in practice, it may not be enough to just reassign the bad eigenvalues, but the eigenvectors of the closed-loop pencil should also be chosen in such a way that they are as well-conditioned as possible (**robust**). This leads to the **robust eigenvalue assignment problem**. We have considered robust eigenvalue assignment of a first-order system in Chapter 11. Here is the statement on the robust eigenvalue assignment of a second-order system.

Robust (Complete) Eigenvalue Assignment for Second-order System

Given

1. Real $n \times n$ matrices $M = M^T > 0$, $D = D^T$, $K = K^T$.
2. The $n \times m$ ($m \leq n$) control matrix B .
3. The self-conjugate number set $S = \{\lambda_{11}, \lambda_{12}, \dots, \lambda_{1n}; \lambda_{21}, \lambda_{22}, \dots, \lambda_{2n}\}$.

Find feedback matrices F and G such that the spectrum of the closed-loop pencil is the set S and the eigenvectors are as well-conditioned as possible.

We next describe some recent algorithms for solving these problems. The feedback stabilization algorithm is **nonmodal**: *it does not require any knowledge of eigenvalues and eigenvectors of the open-loop pencil*.

The algorithms for partial eigenvalue assignment and eigenstructure assignment problems are **partial-modal**: *they require only a partial knowledge of the eigenvalues and eigenvectors.* Specifically, only the small number of eigenvalues that are required to be reassigned and the corresponding eigenvectors are needed. These small number of eigenvalues either can be computed or be measured in a vibration laboratory, given the physical parameters of the system. The single-input partial eigenvalue assignment algorithm (**Algorithm 16.7.2**) is due to Datta, Elhay, and Ram (1997) and its multi-input version (**Algorithm 16.7.3**) appears in Datta, Ram and Sarkissian (2001). We also describe two algorithms for robust eigenvalue assignment problem for second-order system. These algorithms appear in Chu and Datta (1996).

An additional advantage of these algorithms is that they work exclusively with the data matrices M, D , and K of the quadratic pencil, thus allowing the exploitation of structural properties such as symmetry, definiteness, bandness, sparsity, etc., which occur frequently in practical applications.

16.7.3 A Nonmodal Solution of the Feedback Stabilization Problem

Write the closed-loop pencil

$$P_c(\lambda) = \lambda^2 M + \lambda(D - BF^T) + (K - BG^T) = \lambda^2 M + \lambda D' + K'$$

where

$$\begin{aligned} D' &= D - BF^T \\ K' &= K - BG^T. \end{aligned}$$

From the Rayleigh criterion of stability (**Theorem 16.5.5**), we know that the $P_c(\lambda)$ will have all its eigenvalues with negative real parts if F^T and G^T are so chosen that K' and D' are symmetric positive definite.

Assume that the **system is controllable**. Then it can be seen [**Exercise 7**], using the eigenvector criteria of controllability (**Theorem 16.5.1**), that this will happen if F and G are chosen as

$$F^T = -\alpha_1 C_1 B^T, \quad G^T = -\alpha_2 C_2 B^T$$

where C_1 and C_2 are arbitrary $m \times m$ symmetric positive definite matrices, and α_1 and α_2 are arbitrary nonnegative scalars, both not equal to zero.

An alternative way to see this is to use the expression for the real parts of the eigenvalues of the pencil $P_c(\lambda)$.

Let (λ_c, x_c) be an eigenpair of the pencil $P_c(\lambda)$. Let $\lambda_c = \alpha + i\beta$.

Then from Theorem 16.5.3, we have

$$\alpha = \frac{-|\lambda_c|^2 D'_{x_c}}{|\lambda_c|^2 M_{x_c} + K'_{x_c}} = \frac{-|\lambda_c|^2 (D + \alpha_1 BC_1 B^T)_{x_c}}{|\lambda_c|^2 M_{x_c} + (K + \alpha_2 BC_2 B^T)_{x_c}}.$$

Since the system is controllable, we must have $x_c^T B \neq 0$, which implies that $x_c^T BC_1 B^T x_c = (BC_1 B^T)_{x_c} > 0$.

Similarly, it can be seen that $x_c^T BC_2 B^T x_c > 0$.

In addition to assuming $M = M^T > 0, K = K^T$ and $D = D^T$, let's also assume that $K = K^T \geq 0$, and $D = D^T \geq 0$, then we have

$$(D + \alpha_1 BC_1 B^T)_{x_c} = x_c^T (D + \alpha_1 BC_1 B^T) x_c > 0 \text{ and } (K + \alpha_2 BC_2 B^T)_{x_c} = x_c^T (K + \alpha_2 BC_2 B^T) x_c > 0.$$

Thus $\alpha < 0$, proving the result.

The above discussions lead to the following theorem and the algorithm.

Theorem 16.7.1 (Existence of Stabilizing Feedback Solutions)

Given the $n \times n$ symmetric positive semi definite pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ ($M = M^T > 0, K = K^T \geq 0, D = D^T \geq 0$), and the input matrix B of order $n \times m$, there exist a family of feedback matrices F^T and G^T such that the closed-loop pencil

$$P_c(\lambda) = \lambda^2 M + \lambda(D - BF^T) + (K - BG^T)$$

is asymptotically stable, provided that the system is controllable. Explicit expressions for F^T and G^T are given by

$$F^T = -\alpha_1 C_1 B^T, \quad G^T = -\alpha_2 C_2 B^T,$$

where α_1 and α_2 are nonnegative scalars, not both zero, and C_1 and C_2 are arbitrary $m \times m$ symmetric positive definite matrices.

Remarks. In Datta and Rincón (1993), C_1 and C_2 were chosen to be $C_1 = C_2 = (B^T M^{-1} B)^{-1}$. The above generalization is due to Lancaster (1997), who also pointed out that the assumption on D in the above Theorem can be relaxed to $Re(D) \geq 0$, where $Re(D) = \frac{1}{2}(D + D^*)$, if D is complex.

Algorithm 16.7.1 A Nonmodal Stabilizing Algorithm for Second-order System

Input: The matrices M, K , and D with the following properties:

$$M = M^T > 0, K = K^T \geq 0, D = D^T \geq 0.$$

B – The $n \times m$ control matrix

Outputs: Feedback matrices F^T and G^T such that a pencil $P_c(\lambda) = \lambda^2 M + \lambda(D - BF^T) + (K - BG^T)$ is asymptotically stable.

Step 1. Choose two arbitrary $m \times m$ symmetric positive definite matrices C_1 and C_2 .

Step 2. Form

$$\begin{aligned} F^T &= -\alpha_1 C_1 B^T \\ G^T &= -\alpha_2 C_2 B^T \end{aligned}$$

Example 16.7.1

$$M = \begin{pmatrix} 10 & 1 & 1 \\ 1 & 10 & 1 \\ 1 & 1 & 10 \end{pmatrix}, K = \begin{pmatrix} 11 & 1 & 1 \\ 1 & 11 & 1 \\ 1 & 1 & 11 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The eigenvalues of the pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ are $\{-0.1436 \pm 1.0314j, 0 \pm 1.0541j, -0.0323 \pm 1.0530j\}$.

Step 1. Since $m = 1$ choose C_1, C_2 to be any positive numbers. We choose $C_1 = C_2 = 1$.

Step 2. Choose $\alpha_1 = \alpha_2 = 1$.

$$F^T = -b^T = -(1, 0, 0)$$

$$G^T = -b^T = -(1, 0, 0).$$

The eigenvalues of the closed-loop pencil $P_c(\lambda) = \lambda^2 M + \lambda(D - BF^T) + (K - BG^T)$ are $\{-1.1518 \pm 1.0440j, -0.0571 \pm 1.0824j, -0.0179 \pm 1.0569j\}$. **The pencil $P_c(\lambda)$ is asymptotically stable.**

16.7.4 A Direct and Partial Modal Approach for Partial Eigenvalue and Eigenstructure Assignment

In this subsection we develop a **direct partial-modal approach** for the partial eigenvalue and eigenstructure assignment problems, using the orthogonality result proved in Corollary 16.6.1. We start with the single-input partial eigenvalue assignment problem. We first remind the readers of the formal definition of the problem.

Single-input Partial Eigenvalue Assignment in Second-order Control System

The single-input partial eigenvalue assignment problem for a matrix second-order control system is the problem to find the row vectors f^T and g^T such that the spectrum of the closed-loop pencil $P_c(\lambda)$ is $\{\mu_1, \mu_2, \dots, \mu_p; \lambda_{p+1}, \dots, \lambda_{2n}\}$, where $\lambda_1, \lambda_2, \dots, \lambda_p; \lambda_{p+1}, \dots, \lambda_{2n}$ are the eigenvalues of the open-loop $P(\lambda)$ and μ_1, \dots, μ_p are arbitrary numbers.

Theorem 16.7.2 Direct and Partial Modal Solution to Single-input Partial Eigenvalue Assignment in Second-order System.

If $\{\lambda_1, \dots, \lambda_p\} \cap \{\lambda_{p+1}, \dots, \lambda_{2n}\} = \emptyset$ then

(i) For any arbitrary vector β , the feedback vectors f and g defined by

$$f = MX_1\Lambda_1\beta \text{ and } g = -KX_1\beta \quad (16.7.4)$$

are such that $2n-p$ eigenvalues $\lambda_{p+1}, \dots, \lambda_{2n}$ of the closed-loop pencil $P_c(\lambda) = \lambda^2M + \lambda(D-bf^T) + (K-bg^T)$ are the same as those of the open-loop pencil $P(\lambda) = \lambda^2M + \lambda D + K$.

(ii) If pair $(P(\lambda), B)$ is partially controllable with respect to $\Omega_p = \{\lambda_1, \dots, \lambda_p\}$, $0 \notin \Omega_p$ and $\{\mu_1, \dots, \mu_p\} \cap \{\lambda_1, \dots, \lambda_p\} = \emptyset$ then the single-input problem has a unique solution in the above form (16.7.4) with β given by the solution of the linear algebraic system

$$Z_1\beta = (1, 1, \dots, 1)^T, \quad (16.7.5)$$

where

$$Z_1 = \Lambda'_1 Y_1^T M X_1 \Lambda_1 - Y_1^T K X_1. \quad (16.7.6)$$

Proof. The proof of Theorem 16.7.2 comes in two parts. In Part (i), we give a parametric expression for the feedback vectors f and g such that for any arbitrary choice of that parameter (the vector β in (16.7.4)) the eigenvalues of $P(\lambda)$ that are not required to be reassigned remain unchanged. In Part (ii), we show how to choose this vector β such that the closed-loop pencil will contain the eigenvalue set $\{\mu_1, \mu_2, \dots, \mu_p\}$.

Proof of Part (i).

In terms of the eigenvalue and eigenvector matrices, this amounts to proving that

$$MX_2\Lambda_2^2 + (D - bf^T)X_2\Lambda_2 + (K - bg^T)X_2 = 0.$$

To prove this, we consider the eigendecomposition equation again:

$$MX\Lambda^2 + DX\Lambda + KX = 0.$$

From this, we obtain

$$\begin{aligned} & MX_2\Lambda_2^2 + (D - bf^T)X_2\Lambda_2 + (K - bg^T)X_2 \\ &= MX_2\Lambda^2 + DX_2\Lambda_2 + KX_2 - b\beta^T(\Lambda_1 X_1^T M X_2 \Lambda_2 - X_1^T K X_2) \\ &= -b\beta^T(\Lambda_1 X_1^T M X_2 \Lambda_2 - X_1^T K X_2). \end{aligned}$$

Furthermore, from Corollary 16.6.1, we have

$$\Lambda_1 X_1^T M X_2 \Lambda_2 - X_1^T K X_2 = 0.$$

Thus $MX_2\Lambda_2^2 + (D - bf^T)X_2\Lambda_2 + (K - bg^T)X_2 = 0$.

We illustrate Part (i) of Theorem 16.7.2 by the following example.

Example 16.7.2 Let

$$M = \begin{pmatrix} 10 & 1 & 1 \\ 1 & 10 & 1 \\ 1 & 1 & 10 \end{pmatrix}, \quad K = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 7 \end{pmatrix},$$

$$D = \begin{pmatrix} 11 & 2 & 2 \\ 2 & 11 & 2 \\ 2 & 2 & 11 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The eigenvalues of $P(\lambda) = \lambda^2 M + \lambda D + K$ are $\{-0.6028 \pm 0.7476j, 0.0353, -0.0623, -0.9536, -1.0637\}$.

Suppose we would like to change only the eigenvalue 0.0353 keeping the remaining 5 eigenvalues unchanged. Then

$$\Lambda_1 = 0.0353, \quad X_1 = \begin{pmatrix} -0.9267 \\ 0.3049 \\ 0.2169 \end{pmatrix}.$$

Since β can be chosen arbitrarily, we choose $\beta = 15$. This gives

$$f = \begin{pmatrix} -4.6289 \\ 1.2379 \\ 0.8190 \end{pmatrix}, \quad g = \begin{pmatrix} -5.0065 \\ 1.0673 \\ 0.6336 \end{pmatrix}.$$

Verify: The eigenvalues of the closed-loop pencil $P_c(\lambda) = \lambda^2 M + \lambda(D - bf^T) + (K - bg^T)$ are

$$\{-0.6028 \pm 0.7476j, -0.4552, -1.0637, -0.0623, -0.9536\}.$$

Note that 5 remaining eigenvalues of the open-loop pencil (with negative real parts) remain unchanged by this feedback.

Proof of Part (ii).

The proof of (ii) comes in two stages. In stage 1, we show that if the vector β is chosen satisfying (16.7.5), then the feedback vectors f and g defined by (16.7.4) are such that the eigenvalues $\lambda_1, \dots, \lambda_p$ will be assigned to μ_1, \dots, μ_p . In stage 2, we show that the feedback vectors f and g determined this way are real.

To prove stage 1, we note that if there exists a vector β that moves the eigenvalues $\{\lambda_j\}_{j=1}^p$ to the eigenvalue $\{\mu_j\}_{j=1}^p$, then there exists an eigenvector matrix Y_1 of order $n \times p$ consisting of the eigenvectors y_1, \dots, y_p corresponding to the eigenvalues μ_1, \dots, μ_p such that

$$MY_1(\Lambda'_1)^2 + (D - bf^T)Y_1\Lambda'_1 + (K - bg^T)Y_1 = 0,$$

where

$$Y_1 = (y_1, y_2, \dots, y_p), \quad y_j \neq 0, \quad j = 1, 2, \dots, p.$$

Substituting for f, g and rearranging, we have

$$\begin{aligned} MY_1(\Lambda'_1)^2 + DY_1\Lambda'_1 + KY_1 &= b\beta^T(\Lambda_1 X_1^T M Y_1 \Lambda'_1 - X_1^T K Y_1) \\ &= b\beta^T Z_1^T = bc^T, \end{aligned} \tag{16.7.7}$$

where Z_1 is given by (16.7.6) and $c = Z_1\beta$ is a vector that will depend on the scaling chosen for the eigenvectors in Y_1 .

Since the condition (ii) ensures the existence of the set of p vectors $\{y_1, \dots, y_p\}$ such that for each $k = 1, 2, \dots, p$, we have

$$\begin{pmatrix} y_k \\ 1 \end{pmatrix} \in \text{null}(\mu_k^2 M + \mu_k D + K, -b), \quad (16.7.8)$$

we can solve for each of the eigenvectors y_i using the equations

$$(\mu_j^2 M + \mu_j D + K)y_j = b, \quad j = 1, 2, \dots, p \quad (16.7.9)$$

to obtain Y_1 .

This corresponds to choosing the vector $c = (1, 1, \dots, 1)^T$, so, having computed the eigenvectors we could solve the $p \times p$ linear system

$$Z_1\beta = (1, 1, \dots, 1)^T \quad (16.7.10)$$

for β , and hence determine the vectors f and g . Indeed, to prove that Z_1 is nonsingular we subtract

$$\mu_k y_k^T (Mx_j \lambda_j^2 + Dx_j \lambda_j + Kx_j) = 0$$

from

$$(My_k \mu_k^2 + Dy_k \mu_k + Ky_k)^T x_j \lambda_j = b^T x_j \lambda_j$$

and obtain that

$$(\mu_k - \lambda_j)(\mu_k y_k^T Mx_j \lambda_j - y_k^T Kx_j) = b^T x_j \lambda_j.$$

Thus the k, j -th element of the matrix Z_1 is given by

$$z_{kj} = \frac{b^T x_j \lambda_j}{\mu_k - \lambda_j}$$

so

$$Z_1 = C \text{diag}(b^T x_1 \lambda_1, \dots, b^T x_p \lambda_p).$$

The Cauchy matrix $C = \left(\frac{1}{\mu_k - \lambda_j} \right)_{k,j=1}^p$ is nonsingular, since $\{\mu_1, \dots, \mu_p\} \cap \{\lambda_1, \dots, \lambda_p\} = \emptyset$.

Also $b^T x_j \neq 0$ for all $j = 1, \dots, p$, since the pair $(P(\lambda), b)$ is controllable with respect to each λ_j . Thus, Z_1 is nonsingular and the solution of the problem in the form (16.7.4) is unique.

We now prove stage 2.

Since the set $\{\lambda_1, \dots, \lambda_p\}$ is self-conjugate and the coefficient matrices M , D and K of the open-loop pencil $P(\lambda)$ are real, we know that $\lambda_j = \overline{\lambda_k}$ implies that $x_j = \overline{x_k}$ (conjugate eigenvectors correspond to conjugate eigenvalues). Therefore, there exists a nonsingular permutation matrix T such that

$$\overline{X_1} = X_1 T \text{ and } \overline{X_1 \Lambda_1} = X_1 \Lambda_1 T.$$

Similarly, there is a nonsingular permutation matrix T' such that

$$\overline{Y_1} = Y_1 T' \text{ and } \overline{Y_1 \Lambda'_1} = Y_1 \Lambda'_1 T'.$$

Thus, conjugating (16.7.6), we obtain

$$\overline{Z_1} = (T')^T \Lambda'_1 Y_1^T M X_1 \Lambda_1 T - (T')^T Y_1^T K X_1 T = (T')^T Z_1 T,$$

and conjugation of (16.7.10) gives

$$((T')^T Z_1 T) \overline{\beta} = \overline{Z_1 \beta} = (1, 1, \dots, 1)^T = (T')^T (1, 1, \dots, 1)^T,$$

implying that $\overline{\beta} = T^T \beta$.

Therefore,

$$\overline{f} = M(X_1 \Lambda_1 T)(T^T \beta) = f \text{ and } \overline{g} = -K(X_1 T)(T^T \beta) = g$$

which shows that f and g are real vectors. ■

Based on Theorem 16.7.2 we can state the following algorithm.

Algorithm 16.7.2 Direct and Partial-Modal Algorithm for Single-input Partial Eigenvalue Assignment in Second-order System

Inputs:

1. The $n \times n$ matrices M, K , and D ; $M = M^T > 0$, $D = D^T$ and $K = K^T$.
2. The $n \times 1$ control (input) vector b .
3. The set $\{\mu_1, \dots, \mu_p\}$, closed under complex conjugation.
4. The self-conjugate subset $\Omega_p = \{\lambda_1, \dots, \lambda_p\}$ of the open-loop spectrum $\{\lambda_1, \dots, \lambda_p; \lambda_{p+1}, \dots, \lambda_{2n}\}$ and the associated eigenvector set $\{x_1, \dots, x_p\}$.

Outputs: The feedback vectors f and g such that the spectrum of the closed-loop pencil $P_c(\lambda) = \lambda^2 M + \lambda(D - bf^T) + (K - bg^T)$ is $\{\mu_1, \dots, \mu_p, \lambda_{p+1}, \dots, \lambda_{2n}\}$.

Assumptions:

1. The quadratic pencil is partially controllable with respect to the eigenvalues $\Omega_p = \{\lambda_1, \dots, \lambda_p\}$.
2. $\Omega_p \cap \{\lambda_{p+1}, \dots, \lambda_{2n}\} = \emptyset$, $0 \notin \Omega_p$, $\{\mu_1, \dots, \mu_p\} \cap \{\lambda_1, \dots, \lambda_{2n}\} = \emptyset$.

Step 1.

Form $\Lambda_1 = \text{diag}(\lambda_1, \dots, \lambda_p)$ and $X_1 = (x_1, \dots, x_p)$.

Step 2.

Solve for y_1, \dots, y_p :

$$(\mu_j^2 M + \mu_j D + K)y_j = b, \quad j = 1, \dots, p.$$

Step 3.

Form

$$Z_1 = \Lambda'_1 Y_1^T M X_1 \Lambda_1 - Y_1^T K X_1,$$

where $Y_1 = (y_1, \dots, y_p)$ and $\Lambda'_1 = \text{diag}(\mu_1, \dots, \mu_p)$. If Z_1 is ill-conditioned, then warn the user that the problem is ill-posed.

Step 4.

Solve for β :

$$Z_1 \beta = (1, 1, \dots, 1)^T.$$

Step 5.

Form

$$f = M X_1 \Lambda_1 \beta, \quad g = -K X_1 \beta.$$

Example 16.7.3 Consider Example 16.7.2 again.

We will now reassign $\lambda_1 = 0.0353$ to $\mu_1 = -1$; leaving the other eigenvalues invariant. Thus,

$$p = 1, \quad \Lambda'_1 = -1$$

Step 1.

$$\Lambda_1 = 0.0353, \quad X_1 = \begin{pmatrix} -0.9267 \\ 0.3049 \\ 0.2169 \end{pmatrix}.$$

Step 2. Solve for y_1 :

$$(\mu_1^2 M + \mu_1 D + K)y_1 = b$$

giving

$$y_1^T = (-1.500, 0, 0.5000)^T.$$

Step 3.

$$Z_1 = \Lambda'_1 y_1^T M X_1 \Lambda_1 - y_1^T K X_1 = 0.0316.$$

Step 4.

$$\beta = \frac{1}{Z_1} = 31.6598$$

Step 5.

$$f = MX_1\Lambda_1\beta = \begin{pmatrix} -9.7700 \\ 2.6128 \\ 1.7287 \end{pmatrix}$$

$$g = -KX_1\beta = \begin{pmatrix} -10.5671 \\ 2.2527 \\ 1.3373 \end{pmatrix}.$$

Verify. The eigenvalues of the closed-loop pencil $P_c(\lambda) = \lambda^2 M + \lambda(D - bf^T) + (K - bg^T)$ are $\{-0.6028 \pm 0.7476j, -1.0637, -0.0623, -1, -0.9536\}$.

Case 2. Multi-input case

In the multi-input case, we obtain the following generalization of Algorithm 16.7.2. Proof of the algorithm is left for the readers [**Exercise 8**].

Algorithm 16.7.3 *Direct and Partial Modal Algorithm for Multi-input Partial Eigenvalue Assignment in Second-order System*

Inputs:

1. The $n \times n$ matrices M, K , and D ; $M = M^T > 0$, $K = K^T$ and $D = D^T$.
2. The $n \times m$ control (input) matrix B .
3. The set $\{\mu_1, \dots, \mu_p\}$, closed under complex conjugation.
4. The self-conjugate subset $\{\lambda_1, \dots, \lambda_p\}$ of the open-loop spectrum $\{\lambda_1, \dots, \lambda_p; \lambda_{p+1}, \dots, \lambda_{2n}\}$ and the associated eigenvector set $\{x_1, \dots, x_p\}$.

Outputs:

The feedback matrices F and G such that the spectrum of the closed-loop pencil $P_c(\lambda) = \lambda^2 M + \lambda(D - BF^T) + (K - BG^T)$ is $\{\mu_1, \dots, \mu_p, \lambda_{p+1}, \dots, \lambda_{2n}\}$.

Assumptions:

1. The quadratic pencil is (partially) controllable with respect to the eigenvalues $\Omega_p = \{\lambda_1, \dots, \lambda_p\}$.
2. $\Omega_p \cap \{\lambda_{p+1}, \dots, \lambda_{2n}\} = \emptyset$, $0 \notin \Omega_p$, $\{\mu_1, \dots, \mu_p\} \cap \{\lambda_1, \dots, \lambda_{2n}\} = \emptyset$.

Step 1.

Form $\Lambda_1 = \text{diag}(\lambda_1, \dots, \lambda_p)$ and $X_1 = (x_1, \dots, x_p)$.

Step 2.

Choose arbitrary vectors $\gamma_1, \dots, \gamma_p$ in such a way that $\mu_j = \overline{\mu_k}$ implies $\gamma_j = \overline{\gamma_k}$ and solve for y_1, \dots, y_p :

$$(\mu_j^2 M + \mu_j D + K)y_j = B\gamma_j, \quad j = 1, \dots, p.$$

Step 3.

Form

$$Z_1 = \Lambda'_1 Y_1^T M X_1 \Lambda_1 - Y_1^T K X_1,$$

where $Y_1 = (y_1, \dots, y_p)$ and $\Lambda'_1 = \text{diag}(\mu_1, \dots, \mu_p)$. If Z_1 is ill-conditioned, then return to Step 2 and select different vectors $\gamma_1, \dots, \gamma_p$.

Step 4.

Form $\Gamma = (\gamma_1, \gamma_2, \dots, \gamma_p)$ and solve for Φ :

$$\Phi Z_1^T = \Gamma.$$

Step 5.

Form

$$\begin{aligned} F &= M X_1 \Lambda_1 \Phi^T \\ G &= -K X_1 \Phi^T. \end{aligned}$$

Solution to Partial Eigenstructure Assignment Problem

A “direct and partial-modal” solution to the partial eigenstructure assignment problem is now given in the following Theorem.

Theorem 16.7.3 (Direct and Partial Modal Solution to Partial Eigenstructure Assignment in Second-order System)

(i) The triplet $(\hat{B}, \hat{F}, \hat{G})$ defined by

$$\begin{aligned} \hat{B} &= M Y_1 (\Lambda'_1)^2 + D Y_1 \Lambda'_1 + K Y_1, \\ \hat{F} &= M X_1 \Lambda_1 Z_1^{-1}, \text{ and} \\ \hat{G} &= -K X_1 Z_1^{-1} \end{aligned}$$

where

$$Z_1 = \Lambda'_1 Y_1^T M X_1 \Lambda_1 - Y_1^T K X_1,$$

constitutes a (possibly complex) solution to the partial eigenstructure assignment problem, provided that Z_1 is nonsingular.

- (ii)** A solution with real B , F , and G is obtained from the triplet $(\hat{B}, \hat{F}, \hat{G})$ by taking the economy size QR factorization or the singular value decomposition (SVD) of the real matrix $H = \hat{B}(\hat{F}^T, \hat{G}^T)$.

If QR factorization is used and if $LR = H$ is the economy QR factorization of H , then

$$\begin{aligned} B &= L, \\ F^T &= (r_1, r_2, \dots, r_n) \text{ and} \\ G^T &= (r_{n+1}, r_{n+2}, \dots, r_{2n}); \end{aligned}$$

where $R = (r_1, \dots, r_{2n})$.

If SVD is used and if $H = U\Sigma V^T$ is the SVD of H , then the above formulae could be used either with

$$L = U, R = \Sigma V^T,$$

or with

$$L = U\Sigma, R = V^T.$$

Proof of Part (i):

Suppose that the triplet $(\tilde{B}, \tilde{F}, \tilde{G})$ is a solution. Then

$$MY_1(\Lambda'_1)^2 + DY_1\Lambda'_1 + KY_1 = \tilde{B}(\tilde{F}^T Y_1 \Lambda'_1 + \tilde{G}^T Y_1). \quad (16.7.11)$$

Note that if \tilde{B} , \tilde{F} , and \tilde{G} constitute a solution to the problem, then for any invertible W , $\hat{B} = \tilde{B}W$, $\hat{F} = \tilde{F}W^{-T}$, and $\hat{G} = \tilde{G}W^{-T}$ also constitute a solution; because $\tilde{B}\tilde{F}^T = \hat{B}\hat{F}^T$ and $\tilde{B}\tilde{G}^T = \hat{B}\hat{G}^T$. Denote

$$W = \tilde{F}^T Y_1 \Lambda'_1 + \tilde{G}^T Y_1. \quad (16.7.12)$$

Then, provided that W is invertible, $\hat{B} = \tilde{B}W$ is admissible for some \hat{F} and \hat{G} . Thus we can take

$$\hat{B} = MY_1(\Lambda'_1)^2 + DY_1\Lambda'_1 + KY_1 \quad (16.7.13)$$

by virtue of (16.7.11) and (16.7.12). Relation (16.7.13) implies that

$$\hat{F}^T Y_1 \Lambda'_1 + \hat{G}^T Y_1 = I. \quad (16.7.14)$$

Generalizing the result in Part (i) of Theorem 16.7.2, we can show that for any Φ , the matrices

$$\hat{F} = MX_1\Lambda_1\Phi^T \quad \text{and} \quad \hat{G} = -KX_1\Phi^T \quad (16.7.15)$$

satisfy

$$MX_2\Lambda_2^2 + (D - \hat{B}\hat{F}^T)X_2\Lambda_2 + (K - \hat{B}\hat{G}^T)X_2 = 0.$$

Substituting (16.7.15) into (16.7.14), we obtain

$$\Phi = (\Lambda_1 X_1^T M Y_1 \Lambda'_1 - X_1^T K Y_1)^{-1} = (Z_1^T)^{-1}$$

from which \hat{F} and \hat{G} can be determined.

Proof of Part (ii). Since $\mu_j = \overline{\mu_k}$ implies $y_j = \overline{y_k}$, as in the proof of Theorem 16.7.2, we note that there exist permutation matrices T and T' such that

$$\overline{X_1} = X_1 T, \quad \overline{X_1 \Lambda_1} = X_1 \Lambda_1 T, \quad \overline{Y_1} = Y_1 T', \quad \overline{Y_1 \Lambda'_1} = Y_1 \Lambda'_1 T' \text{ and } \overline{Y_1 (\Lambda'_1)^2} = Y_1 (\Lambda'_1)^2 T'.$$

Thus, $\overline{Z_1} = (T')^T Z_1 T$ and using (16.7.13) and (16.7.15) we obtain

$$\begin{aligned} \overline{\hat{B}} &= M Y_1 (\Lambda'_1)^2 T' + D Y_1 \Lambda'_1 T' + K Y_1 T' = \hat{B} T', \\ \overline{\hat{B} \hat{F}^T} &= \hat{B} T' (M X_1 \Lambda_1 T (T^T Z_1^{-1} T'))^T = \hat{B} M X_1 \Lambda_1 Z_1^{-1} = \hat{B} \hat{F}^T \\ \text{and} \quad \overline{\hat{B} \hat{G}^T} &= \hat{B} T' (-K X_1 T (T^T Z_1^{-1} T'))^T = -\hat{B} K X_1 Z_1^{-1} = \hat{B} \hat{G}^T, \end{aligned}$$

which implies that both $\hat{B} \hat{F}^T$ and $\hat{B} \hat{G}^T$ are real matrices.

Define now the real $n \times 2n$ matrix

$$H = \hat{B} (\hat{F}^T, \hat{G}^T)$$

and let $LR = H$ be a factorization of H ; where L and R are, respectively, of order $n \times m$ and $m \times 2n$. Then we can take B to be L , the first n columns of R to be F^T , and the last n columns of R to be G^T . The singular value decomposition of H can be used to compute B , F , and G (see Golub and Van Loan (1996) or Datta (1995)). ■

Based on the Theorem 16.7.3 we can state the following algorithm.

Algorithm 16.7.4 Direct and Partial-Modal Algorithm for Partial Eigenstructure Assignment in Second-order System

Inputs:

1. The $n \times n$ matrices M, K , and D ; $M = M^T > 0$, $K = K^T$, $D = D^T$.
2. The set of p numbers $\{\mu_1, \dots, \mu_p\}$ and the set of p vectors $\{y_1, \dots, y_p\}$, both closed under complex conjugation.
4. The self-conjugate subset $\{\lambda_1, \dots, \lambda_p\}$ of the open-loop spectrum $\{\lambda_1, \dots, \lambda_p; \lambda_{p+1}, \dots, \lambda_{2n}\}$ and the associated eigenvector set $\{x_1, \dots, x_p\}$.

Outputs: The feedback matrices F and G such that the spectrum of the closed-loop pencil $P_c(\lambda) = \lambda^2 M + \lambda(D - BF^T) + (K - BG^T)$ is $\{\mu_1, \dots, \mu_p, \lambda_{p+1}, \dots, \lambda_{2n}\}$ and the eigenvectors corresponding to μ_1, \dots, μ_p are y_1, \dots, y_p , respectively.

Assumptions:

1. $\{\lambda_1, \dots, \lambda_p\} \cap \{\lambda_{p+1}, \dots, \lambda_{2n}\} = \emptyset$.
2. $\mu_j = \overline{\mu_k}$ implies $y_j = \overline{y_k}$ for all $1 \leq j, k \leq p$.
3. The matrix Z_1 obtained in Step 2 is nonsingular.

Step 1. Obtain the first p eigenvalues $\lambda_1, \dots, \lambda_p$ that need to be reassigned and the corresponding eigenvectors x_1, \dots, x_p .

Form $\Lambda_1 = \text{diag}(\lambda_1, \dots, \lambda_p)$, $\Lambda'_1 = \text{diag}(\mu_1, \dots, \mu_p)$, $X_1 = (x_1, \dots, x_p)$ and $Y_1 = (y_1, \dots, y_p)$.

Step 2. Form the matrices \hat{B} and Z_1

$$\begin{aligned}\hat{B} &= MY_1(\Lambda'_1)^2 + DY_1\Lambda'_1 + KY_1, \\ Z_1 &= \Lambda'_1 Y_1^T M X_1 \Lambda_1 - Y_1^T K X_1.\end{aligned}$$

Step 3. Solve for $\hat{H} \in \mathbb{C}^{p \times 2n}$

$$Z_1^T \hat{H} = (\Lambda_1^T X_1^T M, -X_1^T K)$$

and form $H = \hat{B} \hat{H}$.

(If the system does not have a solution, the given set of eigenvectors can not be assigned).

Step 4. Compute a factorization of the real matrix $H = BR$.

Step 5. Partition $R \in \mathbb{R}^{m \times 2n}$ to get $F, G \in \mathbb{R}^{m \times n}$

$$R = (F^T, G^T).$$

(Step 4 can be implemented using the SVD of H).

Note: MATLAB codes for Algorithm 16.7.3 and 16.7.4 are available from the authors upon request.

Illustrative Numerical Examples

Example 16.7.4 We illustrate Algorithm 16.7.3 for the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ with matrices M , D , K and B given by

$$M = \begin{pmatrix} 1.4685 & 0.7177 & 0.4757 & 0.4311 \\ 0.7177 & 2.6938 & 1.2660 & 0.9676 \\ 0.4757 & 1.2660 & 2.7061 & 1.3918 \\ 0.4311 & 0.9676 & 1.3918 & 2.1876 \end{pmatrix}, D = \begin{pmatrix} 1.3525 & 1.2695 & 0.7967 & 0.8160 \\ 1.2695 & 1.3274 & 0.9144 & 0.7325 \\ 0.7967 & 0.9144 & 0.9456 & 0.8310 \\ 0.8160 & 0.7325 & 0.8310 & 1.1536 \end{pmatrix}$$

$$K = \begin{pmatrix} 1.7824 & 0.0076 & -0.1359 & -0.7290 \\ 0.0076 & 1.0287 & -0.0101 & -0.0493 \\ -0.1359 & -0.0101 & 2.8360 & -0.2564 \\ -0.7290 & -0.0493 & -0.2564 & 1.9130 \end{pmatrix} \text{ and } B = \begin{pmatrix} 0.3450 & 0.4578 \\ 0.0579 & 0.7630 \\ 0.5967 & 0.9990 \\ 0.2853 & 0.3063 \end{pmatrix}$$

The open-loop eigenvalues of $P(\lambda)$, computed via MATLAB, are:

$$-0.0861 \pm 1.6242j, -0.1022 \pm 0.8876j, -0.1748 \pm 1.1922j \text{ and } -0.4480 \pm 0.2465j.$$

We reassign only the most unstable pair of the open-loop eigenvalues; namely, $-0.0861 \pm 1.6242j$ to the locations $-0.1 \pm 1.6242j$. That is, we want the closed-loop pencil $P_c(\lambda) = \lambda^2 M + \lambda(D - BF^T) + K - BG^T$ to have the spectrum

$$\{-0.1 \pm 1.6242j, -0.1022 \pm 0.8876j, -0.1748 \pm 1.1922j, -0.448 \pm 0.2465j\}. \quad (16.7.16)$$

The random choices of γ_1 and γ_2 produce matrix Z_1 with the condition number $\text{Cond}_2(Z_1) = \|Z_1\|_2 \|Z_1^{-1}\|_2 = 1.64$ and the feedback matrices

$$F = \begin{pmatrix} 3.3599 & -2.4691 \\ -2.5437 & 1.2692 \\ 10.3080 & -6.8494 \\ -8.0702 & 5.6643 \end{pmatrix} \text{ and } G = \begin{pmatrix} -4.1868 & 1.4318 \\ -0.3352 & 0.4506 \\ -6.4921 & 1.2369 \\ 7.2495 & -2.2698 \end{pmatrix}$$

with $\|F\|_2 = 16.6$ and $\|G\|_2 = 10.99$ such that the spectrum of the closed-loop pencil $P_c(\lambda)$ is precisely the given set. Although Algorithm 16.7.3 is not designed to solve a robust partial eigenvalue assignment problem, it is noted that there are opportunities to exploit the freedom offered by the algorithm to construct possibly robust feedback matrices F and G . While a systematic way to do so is still under investigation, we exploited the freedom of choosing γ_1 and γ_2 in this experiment, yielding the matrices F and G with norms smaller than those as given above.

The method, essentially similar to the method 2/3 in the Kautsky, Nichols and Van Dooren (1985), that uses the freedom in choosing vectors γ_1 and γ_2 in order to improve the condition number of Z_1 , converged after 3 steps, producing the matrix $Z_1^{(\text{robust})}$ with the condition number $\text{Cond}_2(Z_1^{(\text{robust})}) = 1.1$ and the feedback matrices

$$F^{(\text{robust})} = \begin{pmatrix} -0.6532 & 0.4781 \\ 0.7079 & -0.4183 \\ -2.2620 & 1.5349 \\ 1.6636 & -1.1733 \end{pmatrix} \text{ and } G^{(\text{robust})} = \begin{pmatrix} 1.3988 & -0.7501 \\ -0.0075 & -0.0285 \\ 2.5185 & -1.2554 \\ -2.4964 & 1.3185 \end{pmatrix}$$

with the $\|F^{(\text{robust})}\|_2 = 3.6$ and $\|G^{(\text{robust})}\|_2 = 4.3$ and the spectrum of the “robust” closed-loop pencil

$$P_c^{(\text{robust})}(\lambda) = \lambda^2 M + \lambda \left(D - B \left(F^{(\text{robust})} \right)^T \right) + \left(K - B \left(G^{(\text{robust})} \right)^T \right)$$

is precisely the given set (16.7.16).

We call the last closed-loop pencil “robust” because, aside from the mere reduction in the norm of the feedback matrices, our numerical experiments suggest that the eigenvalues of $P_c^{(\text{robust})}(\lambda)$ are less affected by the random perturbations of feedback matrices. This is illustrated in the Figure below that plots the convex hulls of the closed-loop eigenvalues, when the feedback

matrices F , G , $F^{(\text{robust})}$ and $G^{(\text{robust})}$ are perturbed, respectively, by ΔF , ΔG , $\Delta F^{(\text{robust})}$ and $\Delta G^{(\text{robust})}$, such that

$$\|\Delta F\|_2 < 0.01\|F\|_2, \|\Delta G\|_2 < 0.01\|G\|_2$$

and

$$\|\Delta F^{(\text{robust})}\|_2 < 0.01\|F^{(\text{robust})}\|_2, \|\Delta G^{(\text{robust})}\|_2 < 0.01\|G^{(\text{robust})}\|_2,$$

using 200 random perturbations.

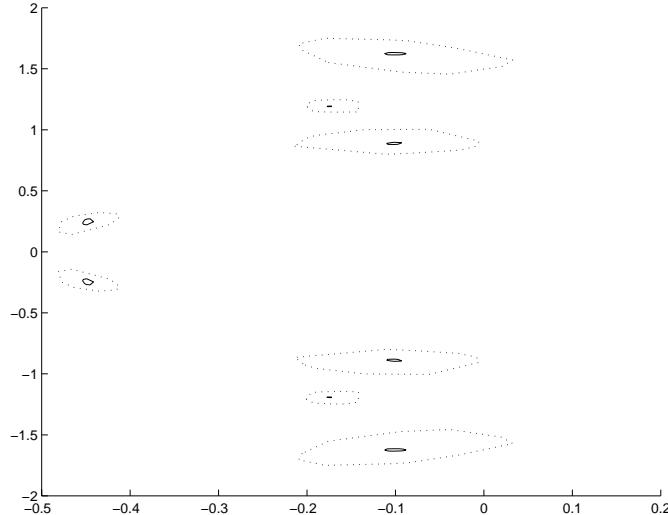


Figure 16.1: The convex hulls of closed-loop eigenvalues under 200 random 1% perturbation of the feedback matrices for quadratic pencils $P_c^{(\text{robust})}(\lambda)$ (solid lines) and $P_c(\lambda)$ (dashed lines).

Example 16.7.5 The eigenstructure assignment problem is now solved, reassigning again the most unstable pair of the open-loop eigenvalues; namely, $-0.0861 \pm 1.6242j$ to the same locations $-0.1 \pm 1.6242j$. That is, we want the closed-loop pencil to have the spectrum (16.7.16). Let the matrix of vectors to be assigned be:

$$Y_1 = \begin{pmatrix} 1.0000 & 1.0000 \\ 0.0535 + 0.3834j & 0.0535 - 0.3834j \\ 0.5297 + 0.0668j & 0.5297 - 0.0668j \\ 0.6711 + 0.4175j & 0.6711 - 0.4175j \end{pmatrix}.$$

Algorithm 16.7.4 produced the control matrix

$$B = \begin{pmatrix} -0.3814 & -0.5751 \\ -0.5555 & -0.4821 \\ -0.5191 & 0.4311 \\ -0.5258 & 0.5010 \end{pmatrix}$$

with $\|B\|_2 = 1$ and the feedback matrices

$$F = \begin{pmatrix} -8.1693 & -0.2320 \\ 2.5326 & -0.4130 \\ -20.6466 & 0 \\ 18.0019 & 0.2962 \end{pmatrix} \text{ and } G = \begin{pmatrix} 0.1688 & -1.3245 \\ 2.0584 & 0.2236 \\ -5.7223 & -3.0184 \\ 0.9821 & 2.4989 \end{pmatrix}$$

with the norms $\|F\|_2 = 28.6976$ and $\|G\|_2 = 7.07673$. The spectrum of the closed-loop pencil $P_c(\lambda)$ with above F and G is precisely the set (16.7.16) and the eigenvectors, corresponding to the eigenvalues $-0.1 \pm 1.6242j$ are the columns of Y_1 .

Remark 16.7.1. Examples 16.7.4 and 16.7.5 are purely illustrative ones. A real-life example involving an 211×211 quadratic pencil with sparse matrices has been solved in (Datta and Sarkissian (1999)), using Algorithm 16.7.3.

Remark 16.7.2 The robustness of the closed-loop eigenvalues has been achieved by choosing the parametric vectors γ_1 and γ_2 appropriately.

Another idea to achieve robustness would be to select the poles appropriately in a certain region of the complex plane. In recent papers Calvetti et al. (Calvetti, Lewis and Reichel (1998), (1999)), have shown that the idea works quite well in the single-input case for a first-order state-space system.

16.7.5 Robust Eigenvalue Assignment in Second-order System

In this section, we present two algorithms for **robust eigenvalue assignment problem** that can be implemented directly in second-order setting.

Algorithm 16.7.5 is a **modification** of an algorithm by Juang and Maghami (1992) which is an adaptation to the second-order model of the well-known first order algorithm for robust

eigenvalue assignment by Kautsky, Nichols, and Van Dooren (1985), described earlier in Chapter 11.

Algorithm 16.7.6 is a generalization of the feedback stabilization Algorithm 16.7.1.

We state Algorithm 16.7.5 without proof. The proof is left as an exercise [**Exercise 9**].

Algorithm 16.7.6 is developed in details.

In this Subsection, *we use different notations than what we have used in the previous sections*. Here the closed-loop eigenvalues are denoted by $\lambda_{11}, \lambda_{12}, \dots, \lambda_{1n}; \lambda_{21}, \lambda_{22}, \dots, \lambda_{2n}$. Also, Λ_1 and Λ_2 here are defined as:

$$\begin{aligned}\Lambda_1 &= \text{diag}(\lambda_{11}, \lambda_{12}, \dots, \lambda_{1n}), \\ \Lambda_2 &= \text{diag}(\lambda_{21}, \lambda_{22}, \dots, \lambda_{2n}).\end{aligned}$$

The matrix of the eigenvectors of the closed-loop pencil is denoted by X_1 . Moreover, the closed-loop pencil $P_c(\lambda)$ here is taken in the form:

$$P_c(\lambda) = \lambda^2 M + \lambda(D + BF^T) + (K + BG^T)$$

Algorithm 16.7.5 Robust Eigenvalue Assignment in Second-order System (Algorithm I)

Inputs:

- (i) *The matrices M, K , and D of a second-order system with the properties: $M = M^T > 0$, $K = K^T$, $D = D^T$.*
- (ii) *The self conjugate set $\{\lambda_{11}, \dots, \lambda_{1n}, \lambda_{21}, \dots, \lambda_{2n}\}$.*
- (iii) *The $n \times m$ full-rank control matrix B .*

Outputs: *The feedback matrices F and G such that the eigenvalues of the closed-loop pencil*

$$P_c(\lambda) = \lambda^2 M + \lambda(D + BF^T) + (K + BG^T)$$

are $\{\lambda_{11}, \lambda_{12}, \dots, \lambda_{1n}, \lambda_{21}, \lambda_{22}, \dots, \lambda_{2n}\}$.

Assumption: *The pair $(P(\lambda), B)$ is controllable.*

Step 1. (Initialization) Set $\Lambda_i \equiv \text{diag}(\lambda_{i1}, \dots, \lambda_{in})$, $i = 1, 2$.

Form the QR decomposition of B , so that

$$B = (Q_1, Q_2) \begin{pmatrix} R_B \\ 0 \end{pmatrix} = Q_1 R_B, \quad Q = (Q_1, Q_2),$$

with R_B being upper triangular and nonsingular, and Q being orthogonal.

Step 2. (Eigenvector Selection)

For $i = 1, 2$ and $k = 1, \dots, n$, select x_{ik} from the null spaces (using QR decompositions) defined by

$$Q_2^T(\lambda_{ik}^2 M + \lambda_{ik} D + K)x_{ik} = 0 ,$$

so as to minimize the condition number of

$$S_1 = \begin{pmatrix} X_1 \\ X_1 T_1 \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} \\ X_{11}\Lambda_1 & X_{12}\Lambda_2 \end{pmatrix},$$

where $X_{1i} = (x_{i1}, \dots, x_{in})$ $i = 1, 2$, $X_1 = (X_{11}, X_{12})$ and $T_1 = \Lambda_1 \oplus \Lambda_2$.

Step 3. (Computations of Feedback Matrices)

Solve the well-conditioned system

$$R_B(F^T, G^T)S_1 = -Q_1^T(R_1, R_2),$$

for the feedback matrices G^T and F^T , with $R_i = MX_{1i}\Lambda_i^2 + DX_{1i}\Lambda_i + KX_{1i}$ ($i = 1, 2$).

Remarks:

1. The eigenvector selection in Step 2 can be done in a similar fashion as in Kautsky, Nichols, and Van Dooren (1985). Note that from the assumed controllability property of the system, we can deduce easily that the dimensions of the null spaces defined by Step 2 are exactly m . As a result, we can only assign eigenvalues of multiplicity less than or equal to m , otherwise a more involved procedure involving defective (thus ill-conditioned) eigenvalues and principal vectors will be required. On the other hand, open-loop eigenvalues can be re-assigned with different eigenvectors. As indicated in Step 2, it is possible to select x_{ik} so that the condition number of S_1 is optimized.
2. In Juang and Maghami (1990), it was indicated that only n eigenvalues are assignable, but we have seen above that it is possible to assign all the $2n$ eigenvalues while minimizing the condition number of the eigenvector matrix.

Development of Algorithm 16.7.6

Let X_1 be the same as in the previous algorithm.

Assume that X_{11} , the first n columns in X_1 , is nonsingular. With $C_0 \equiv X_{11}\Lambda_1 X_{11}^{-1}$, $D_0 \equiv \tilde{Y}\Lambda_2\tilde{Y}^{-1}$ and $\tilde{Y} \equiv (\tilde{y}_1, \dots, \tilde{y}_n)$, define

$$L_0 \equiv \begin{pmatrix} I & 0 \\ C_0 & -I \end{pmatrix}, \quad \tilde{T}_1 = \begin{pmatrix} C_0 & -I \\ 0 & D_0 \end{pmatrix}.$$

Obviously, \tilde{T}_1 has the same spectrum as $T_1 = \text{diag}(\Lambda_1, \Lambda_2)$, and $L_0^2 = I$. Using the first order linearization $A = \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix}$, we have

$$A - L_0^{-1}\tilde{T}_1L_0 = \begin{pmatrix} 0 & 0 \\ -M^{-1}K + D_0C_0 & -M^{-1}D - (C_0 + D_0) \end{pmatrix}.$$

Re-arranging and comparing with the closed-loop linearization

$$A - \begin{pmatrix} 0 & 0 \\ M^{-1}BG^T & M^{-1}BF^T \end{pmatrix},$$

we have the **sufficient** (but not necessary) conditions for G^T and F^T assigning the eigenvalues in T_1 :

$$BG^T = -K + MD_0C_0,$$

and

$$BF^T = -D - M(C_0 + D_0).$$

The feedback matrices G^T and F^T can be obtained via the solution of these two equations, possibly in the least squares sense, for given C_0 and D_0 . Unfortunately, these equations are consistent only for suitably chosen D_0 and C_0 (see later), and the least squares solution does not guarantee the assignment, approximate or otherwise, of the prescribed closed-loop poles. We now show how the above approach can be made to work.

From the last equation we have the expression, $MD_0 = -D - BF^T - MC_0$. Substituting this expression of D_0 into $BG^T = -K + MD_0C_0$, we have

$$B(G^T + F^T C_0) = -(K + DC_0 + MC_0^2).$$

Pre-multiplying by Q_2^T and then post-multiplying by x_{1k} , and noting that $C_0 = X_{11}\Lambda_1X_{11}^{-1}$, we have

$$Q_2^T(\lambda_{1k}^2 M + \lambda_{1k} D + K)x_{1k} = 0, \quad k = 1, \dots, n.$$

With $X_{11} = (x_{11}, \dots, x_{1n})$ selected from the null spaces defined as above, C_0 can then be calculated. Pre-multiplying $B(G^T + F^T C_0) = -(K + DC_0 + MC_0^2)$, by $B^\dagger = R_B^{-1}Q_1^T$ gives rise to

$$G^T = -F^T C_0 - R_B^{-1}Q_1^T(K + DC_0 + MC_0^2),$$

from which G^T can be calculated in terms of F^T . However, a similar but simpler formula for G^T can be obtained by pre-multiplying by B^\dagger , so that

$$G^T = R_B^{-1}Q_1^T(-K + MD_0C_0),$$

which is applicable after both C_0 and D_0 are obtained. A similar formula for F^T can be obtained from using the same technique, so that

$$F^T = -R_B^{-1}Q_1^T(D + M(C_0 + D_0)).$$

To obtain $D_0 \equiv \tilde{Y}\Lambda_2\tilde{Y}$, pre-multiply $BF^T = -D - M(C_0 + D_0)$ by Q_2^T and post-multiply by \tilde{y}_k ($k = 1, \dots, n$) to produce

$$Q_2^T(-D - MC_0 - \lambda_{2k}M)\tilde{y}_k = 0, \quad Q_2^TBF^T\tilde{y}_k = 0, \quad k = 1, \dots, n.$$

Columns in \tilde{Y} can thus be selected from the null spaces defined by above, after C_0 has been calculated.

**Algorithm 16.7.6 Robust Eigenvalue Assignment in Second-order System
(Algorithm II)**

Inputs:

- (i) The matrices M, K , and D of second-order system with properties:

$$M = M^T > 0, \quad K = K^T, \quad D = D^T.$$

- (ii) The self-conjugate set $\{\lambda_{11}, \lambda_{12}, \dots, \lambda_{1n}; \lambda_{21}, \dots, \lambda_{2n}\}$.
- (iii) The $n \times m$ full-rank control matrix B .

Outputs: Feedback matrices F and G such that the eigenvalues of the closed-loop pencil

$$P_c(\lambda) = \lambda^2 M + \lambda(D + BF^T) + F(K + BG^T)$$

are $\{\lambda_{11}, \dots, \lambda_{1n}; \lambda_{21}, \dots, \lambda_{2n}\}$.

Assumptions: The pair $(P(\lambda), B)$ is controllable.

Step 1. Partition the eigenvalues: $\Lambda_i \equiv \text{diag}(\lambda_{i1}, \dots, \lambda_{in}), i = 1, 2$.

Step 2.

Form the QR decomposition of B , so that

$$B = (Q_1, \quad Q_2) \begin{pmatrix} R_B \\ 0 \end{pmatrix} = Q_1 R_B, \quad Q = (Q_1, \quad Q_2),$$

where R_B is upper triangular and nonsingular, and Q is orthogonal.

Step 3. (Eigenvector Selection for X_{11}). For $k = 1, \dots, n$, select x_{ik} from the null spaces (realized using QR decompositions) defined by:

$$Q_2^T (\lambda_{1k}^2 M + \lambda_{1k} D + K) x_{1k} = 0,$$

so that $X_{11} = (x_{11}, \dots, x_{1n})$ is well-conditioned.

Solve $C_0 X_{11} = X_{11} \Lambda_1$ for C_0 .

Step 4. (Eigenvector Selection for \tilde{Y}). For $k = 1, \dots, n$, select \tilde{y}_k from the null spaces defined by:

$$Q_2^T (-D - MC_0 - \lambda_{2k} M) \tilde{y}_k = 0,$$

so that $\tilde{Y} = [\tilde{y}_1, \dots, \tilde{y}_n]$ is well-conditioned.

Step 5. (Calculation of Feedback Matrices). Solve for G^T and F^T :

$$R_B(G^T, F^T) = Q_1^T (-K + MD_0C_0, -D - M(C_0 + D_0)).$$

Remarks: There are several **differences and similarities** between the closely related Algorithms 16.7.5 and 16.7.6. Firstly, the assumption that X_{11} and \tilde{Y} are nonsingular is required for Algorithm 16.7.6, but not for Algorithm 16.7.5. From the theory of matrix polynomials (see Chu (1994)), S_1 is always nonsingular, implying that X_1 always has full rank. **As a result, it is always possible to find a partition** $T_1 = \Lambda_1 \oplus \Lambda_2$ such that X_{11} , which contains as columns the eigenvectors corresponding to Λ_1 , is nonsingular. However, this partition is not known in advance and we may have to repeat our calculations when X_{11} is detected to be ill-conditioned. There is a slight advantage with Algorithm 16.7.5 that the selection of $\{x_{1j}\}$ and $\{x_{2k}\}$ can be performed in parallel, in contrast with the fact that X_{11} and C_0 have to be determined before the selection of $\{\tilde{y}_k\}$ in Algorithm 16.7.6.

We shall now elaborate on the left-eigenvectors. Let the rows of Y be the left-eigenvectors of closed-loop system matrix, for given feedback matrices G^T and F^T , of the linearized standard first-order closed-loop system with corresponding eigenvalues on the diagonal of $T_1 = \Lambda_1 \oplus \Lambda_2$. Let $Y = S_0^{-1}$ and

$$Y = [Y_1, Y_2] = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}.$$

Then we can show that $M\tilde{Y} = Y_{22}^{-1}$, implying that \tilde{Y} will be nonsingular for a suitable partition $T_1 = \Lambda_1 \oplus \Lambda_2$ (which yields a nonsingular Y_{22}). It remains an open question whether the controllability of or more generally the solvability of the pole assignment problem, guarantee the existence of a partition $T_1 = \Lambda_1 \oplus \Lambda_2$ which implies the invertibility of both X_{11} and \tilde{Y} .

Finally in Steps 3 and 4 for Algorithm 16.7.6, the eigenvectors can be selected in a similar fashion as in Kautsky, Nichols and Van Dooren (1985). Similar to Algorithm 16.7.5, the controllability property implies that the dimensions of the null-spaces defined in Steps 3 and 4 are exactly m . We do not know much about these null-spaces except that they are not empty and generically should be of dimension m as well. Also, alternatives to minimizing the condition numbers of X_{11} and \tilde{Y} will be discussed in the next section.

Eigenvector Selection

Recall that the condition number of a matrix Z is denoted by $Cond(Z) \equiv \|Z\| \|Z^\dagger\|$, where Z^\dagger is the pseudo inverse of Z . The symbol $\|\cdot\|$ denotes the 2-norm or Frobenius norm, unless otherwise stated.

From the Bauer-Fike Theorem, (Chu (1994)), it is known that

$$Cond(P) \equiv \|X_1\| \|Y_2\|$$

is a condition number for the whole spectrum, without any restriction on the sizes of the perturbations to the original data of M , D and K .

It is difficult to select eigenvectors to minimize $Cond(P)$. However, basic matrix inequalities imply $Cond(P) \leq \|X_1\| \|S_0^{-1}\| \leq Cond(S_0) \leq Cond(\tilde{M}) Cond(S_1)$, $\tilde{M} = I \oplus M$, where

$$S_0 = \tilde{M}S_1 = \begin{pmatrix} X_1 \\ MX_1T_1 \end{pmatrix}.$$

These inequalities give rise to the possibilities of minimizing various upper bounds of $\text{Cond}(P)$. When the right-eigenvectors are normalized, we may minimize the simpler quantities in

$$\|Y_2\| \leq \|S_0^{-1}\| \leq \|\tilde{M}^{-1}\| \|S_1^{-1}\|.$$

For Algorithm 16.7.6, **the above quantities are not available from the computations directly**, and it **may be more efficient to minimize the related quantities which have to be inverted**, e.g., $\text{Cond}(X_{11})$, $\text{Cond}(\tilde{Y})$, $\|X_{11}^{-1}\|$ or $\|\tilde{Y}^{-1}\|$.

For both Algorithm 16.7.5 and 16.7.6, another alternative is to choose a few randomly chosen eigenvectors, evaluate the corresponding condition numbers (or related upper bounds), and then select the best sub-optimal solution. *This “random-search” approach seems to be adequate and yet especially powerful for large control systems with sparsity in M , D , K and B .* For such systems, the full iterative approach for minimizing condition numbers may be prohibitively expensive, especially when only the most ill-conditioned solutions need to be avoided. We shall consider the full iterative approach in what follows.

Let us now reiterate a technique from Kautsky, Nichols, and Van Dooren (1985) for the minimization of $\text{Cond}(X)$ in the Frobenius norm, by updating the j th column x_j in $X = (X_j, x_j)$ (see **Chapter 11**). Notice that we have rearranged the columns in X so that the j th column appears last, without loss of generality and for simplicity in notation. The technique can be applied in the minimization of $\text{Cond}(X_{11})$, $\text{Cond}(\tilde{Y})$, $\|X_{11}^{-1}\|$ or $\|\tilde{Y}^{-1}\|$ for Algorithm 16.7.5 and 16.7.6.

Consider the QR decomposition of

$$X_j = (Q_j, q_j) \begin{pmatrix} R_j \\ 0^T \end{pmatrix} = Q_j R_j,$$

where R_j is upper triangular and assumed to be nonsingular, and $\tilde{Q} \equiv (Q_j, q_j)$. We then have

$$X = (X_j, x_j) = \tilde{Q} \begin{pmatrix} R_j & Q_j^* x_j \\ 0^T & q_j^* x_j \end{pmatrix}.$$

Note that

$$X^{-1} = \begin{pmatrix} R_j^{-1} & -\alpha_j^{-1} R_j^{-1} Q_j^* x_j \\ 0 & \alpha_j^{-1} \end{pmatrix} \tilde{Q}^*, \quad \alpha_j \equiv q_j^* x_j.$$

Assuming that the columns of X are normalized to unity, minimizing $\text{Cond}(X)$ is then equivalent to $\min_{x_j} \|X^{-1}\|$ or

$$\min_{x_j} \left\| \alpha_j^{-1} \begin{pmatrix} R_j^{-1} Q_j^* \\ I \end{pmatrix} x_j \right\|.$$

Let N_j be a unitary matrix with columns spanning the null-space from which x_j should be selected. Thus $x_j = N_j u_j$ with $\|u_j\| = 1$. Form the QR decomposition

$$N_j^T q_j = \hat{Q}_j \begin{pmatrix} \beta_j \\ 0 \end{pmatrix} = \beta_j \hat{Q}_j e_1,$$

where e_j is the j th column of the identity matrix. Consequently, we have $q_j^* N_j \hat{Q}_j = \bar{\beta}_j e_1^T$, $\alpha_j = (q_j^* N_j \hat{Q}_j)(\hat{Q}_j^* u_j) = \bar{\beta}_j e_1^T (\hat{Q}_j^* u_j)$, and the minimization problem reduces to (after ignoring β_j)

$$\min_{\hat{u}_j} \left\| \begin{pmatrix} R_j^{-1} Q_j^* N_j \\ I \end{pmatrix} \hat{Q}_j \begin{pmatrix} \hat{Q}_j^* u_j \\ e_1^T (\hat{Q}_j^* u_j) \end{pmatrix} \right\| .$$

Let

$$\frac{\hat{Q}_j^* u_j}{e_1^T (\hat{Q}_j^* u_j)} = \begin{pmatrix} 1 \\ \hat{u}_j \end{pmatrix}, \quad \hat{u}_j \equiv \begin{pmatrix} \hat{u}_{j2} \\ \vdots \\ \hat{u}_{jn} \end{pmatrix} .$$

We then have to solve the equivalent minimization problem

$$\min_{\hat{u}_j} \left\| \begin{pmatrix} R_j^{-1} Q_j^* N_j \\ I \end{pmatrix} \hat{Q}_j \begin{pmatrix} 1 \\ \hat{u}_j \end{pmatrix} \right\| = \min_{\hat{u}_j} \|Z_j \hat{u}_j + z_j\| ,$$

which is a standard least squares problem and can be solved again by the QR decomposition of Z_j , with $\hat{u}_j = -Z_j^\dagger z_j$. The eigenvector x_j can be recovered from z_j .

When an eigenvalue to be assigned is complex, the above rank-1 update technique will no longer be adequate, as two eigenvectors corresponding to the complex conjugate pair of eigenvalues have to be selected simultaneously. A rank-2 update technique can easily be developed from the procedure described above. The easy-to-solve least squares problem will then become a minimization problem involving a quadratic pencil.

The minimization of $\text{Cond}(S_1)$ can then be carried out as follows:

Recall that

$$S_1 = \begin{pmatrix} X_1 \\ X_1 T_1 \end{pmatrix} .$$

The j th column of S_1 , z_j , satisfies

$$\|z_j\|_2^2 = (1 + |\lambda_{ik}|^2) \|x_{ik}\|_2^2 ,$$

for some values of i and k with $j = (i-1)n+k$. Assume that the columns in X_1 are normalized to unity. As $1 + |\lambda_{ik}|^2$ is constant for the purpose of the minimization, the process is then equivalent to $\min \|S_1^{-1}\|$. With

$$S_1 = \begin{pmatrix} X_j & x_j \\ X_j \Lambda_j & \lambda_{ik} x_j \end{pmatrix} , \quad T_1 = \Lambda_j \oplus \lambda_{ik} ,$$

$$\begin{pmatrix} X_j \\ X_j \Lambda_j \end{pmatrix} = \tilde{Q} \begin{pmatrix} R_j \\ 0^T \end{pmatrix} , \quad \tilde{Q} \equiv (Q_j, q_j) ,$$

we have

$$S_1 = \tilde{Q} \begin{pmatrix} Q_j^* R_j & Q_j^* Z_j x_j \\ 0^T & q_j^* Z_j x_j \end{pmatrix} , \quad Z_j \equiv \begin{pmatrix} I \\ \lambda_{ik} I \end{pmatrix} .$$

The above rank-1 update procedure can then be generalized in a straightforward manner to update columns of S_1 . Similar techniques can obviously be applied to $\min \|S_0^{-1}\|$, with Z_j replaced by $\begin{pmatrix} I \\ \lambda_{ik}M \end{pmatrix}$.

Example 16.5.6. Let

$$M = \text{diag}(10, 10, 10), \quad K = \begin{pmatrix} 40 & -40 & 0 \\ -40 & 80 & -40 \\ 0 & -40 & 80 \end{pmatrix}, \quad D = 0, \quad B = \begin{pmatrix} 1 & 2 \\ 3 & 2 \\ 3 & 4 \end{pmatrix},$$

$$\Lambda_1 = \text{diag}(-1, -2, -3), \quad \Lambda_2 = \text{diag}(-4, -5, -6).$$

Λ_i ($i = 1, 2$) are chosen arbitrarily. The open-loop eigenvalues are

$$\{\pm 3.6039j, \pm 2.4940j, \pm 0.8901j\}.$$

As $p = 2$, the solution is not unique. **A full minimization in Step 2 of Algorithm 16.7.5 or Steps 3 and 4 of Algorithm 16.7.6 has not been performed.** Instead, three combinations (cases (a), (b), and (c)) of eigenvectors from the null-spaces in Step 2 of Algorithm 16.7.5 and in Step 3 and Step 4 of Algorithm 16.7.6, producing several feedback matrices, were considered. In cases (a) and (c), all the eigenvectors were selected, respectively, from the first and last vectors in the null-spaces. In case (b), the average of the two vectors in the null-spaces is selected. The results for cases (a)–(c) from Algorithm 16.7.5 and 16.7.6 are summarized respectively in the enclosed Tables. In these tables, columns 2 and 3 contain various quantities from Algorithm 16.7.5 and 16.7.6. Note that it will be difficult to compare the two methods, as the solutions from them correspond to different closed-loop eigenvectors.

For case (a) in Table 16.1, Algorithm 16.7.5 gave rise to a more well-conditioned solution, in the sense that the condition numbers are slightly smaller. Interestingly, the closed-loop eigenvalues from Algorithm 16.7.6 are slightly more accurate. It is worth noting that the condition numbers usually are conservative upper bounds, and large condition numbers are necessary but not sufficient for an ill-conditioned problem. Again, it is meaningless to compare the two solutions with different eigenvectors. The Example itself is obviously ill-conditioned, with such large condition numbers, although the closed-loop eigenvalues are still accurate enough (**up to eight significant figures**).

For case (b) in Table 16.2, Algorithm 16.7.5 is definitely superior compared to Algorithm 16.7.6, producing a smaller condition number for the solution, smaller gain for the feedback matrices, as well as closed-loop eigenvalues which are at least a hundred times more accurate. The solution from Algorithm 16.7.5 is accurate enough for most applications, with up to seven significant figures in the closed-loop eigenvalues. The solution from Algorithm 16.7.6 is only accurate up to five significant figures. The real and imaginary parts of a complex number are denoted by $Re(\lambda)$ and $Im(\lambda)$.

Table 16.1

	Algorithm 16.7.5		Algorithm 16.7.6	
G^T	-3.3291×10^3	2.6273×10^3	1.0105×10^4	-8.6740×10^3
	-2.9716×10^4	2.3555×10^4	1.2150×10^4	-1.0392×10^4
	6.8797×10^4	-5.4501×10^4	-5.1170×10^4	4.3855×10^4
F^T	-8.6617×10^3	6.8633×10^3	1.9502×10^2	-1.9141×10^2
	2.7906×10^4	-2.2112×10^4	9.7301×10^2	-8.5292×10^2
	-3.3444×10^4	2.6500×10^4	-1.4141×10^3	1.3132×10^3
$Cond(P)$	2.1544×10^6		8.1493×10^6	
$Cond(S_0)$	7.3384×10^6		3.2436×10^7	
$Re(\lambda)$	-6.000000018342428		-5.999999999607684	
	-4.99999948895886		-5.000000001614143	
	-4.000000055236776		-3.99999997502286	
	-2.99999969681889		-3.000000001794087	
	-2.000000008120103		-1.99999999371852	
	$-9.99999992679218 \times 10^{-1}$		-1.000000000095840	
$Im(\lambda)$	$O(10^{-11})$		$O(10^{-11})$	

Table 16.2

	Algorithm 16.7.5		Algorithm 16.7.6	
G^T	-4.6181×10^4	2.8661×10^4	-9.7709×10^5	8.0021×10^5
	-1.2857×10^5	7.9758×10^4	-1.7775×10^6	1.4555×10^6
	1.5868×10^4	-9.8317×10^3	3.1970×10^3	-2.5930×10^3
F^T	5.7132×10^4	-3.5433×10^4	9.4011×10^3	-6.6629×10^3
	1.1536×10^5	-7.1543×10^4	3.3083×10^3	-8.2563×10^2
	-4.1116×10^3	2.5500×10^3	-2.9920×10^1	2.1227×10^1
$Cond(P)$	1.1410×10^7		9.9619×10^9	
$Cond(S_0)$	1.9647×10^7		3.0617×10^{10}	
$Re(\lambda)$	-5.99999946546182		-5.999996413586461	
	-5.000000166766821		-5.000014093971046	
	-3.999999805231928		-3.999978091004949	
	-3.000000107589290		-3.000017029943044	
	-1.99999970741010		-1.999993225845674	
	-1.000000003037593		-1.000001146454624	
$Im(\lambda)$	$O(10^{-13})$		$O(10^{-8})$	

Table 16.3

	Algorithm 16.7.5		Algorithm 16.7.6	
G^T	1.3564×10^4	2.5644×10^3	-6.5140×10^5	4.5089×10^5
	1.3933×10^4	2.5203×10^3	-6.3163×10^5	4.3729×10^5
	-1.4171×10^4	-2.5154×10^3	6.4590×10^5	-4.4716×10^5
F^T	-5.8738×10^3	-1.0347×10^3	-4.6153×10^3	3.6042×10^3
	-7.8963×10^3	-1.3910×10^3	6.3244×10^3	-3.9812×10^3
	7.5902×10^3	1.3371×10^3	-6.3401×10^3	3.9830×10^3
$Cond(P)$	7.9887×10^6		1.8077×10^9	
$Cond(S_0)$	1.2204×10^7		5.5354×10^9	
$Re(\lambda)$	-5.999999991388784		-5.999999615792164	
	-5.000000018032756		-5.000001557116362	
	-3.999999990186220		-3.999997525600821	
	-2.999999997226952		-3.000001935263001	
	-2.000000003539084		-1.999999235603004	
	$-9.999999992460400(-1)$		-1.000000128258480	
$Im(\lambda)$	$O(10^{-15})$		$O(10^{-9})$	

16.8 Summary and Review

Vibrating structures such as buildings, bridges, highways, air and space crafts, etc., are usually modeled by a system of matrix second-order differential equations of the form:

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = 0,$$

which are finite-element generated reduced-order approximations of natural mathematical models of distributed parameter systems.

This Chapter is devoted to the study of control, with a special attention to numerically solving feedback control problems, in matrix second-order systems.

Stability and nature of vibrations of second-order systems are governed by the eigenvalues and eigenvectors of the associated quadratic matrix pencil

$$P(\lambda) = \lambda^2 M + \lambda D + K.$$

I. Quadratic Eigenvalue Problem

Numerical methods for quadratic eigenvalue problem $P(\lambda)x = 0$ are not well-developed. A brief discussion is included in **Section 16.3**. Two natural ways of solving the quadratic eigenvalue problems are via (i) reduction of the problem to a standard eigenvalue problem of the form

$$\begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix} \begin{pmatrix} x \\ \lambda x \end{pmatrix} = \lambda \begin{pmatrix} x \\ \lambda x \end{pmatrix}$$

and (ii) symmetric generalized eigenvalue problem of the type

$$\lambda \begin{pmatrix} -K & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} x \\ \lambda x \end{pmatrix} = \begin{pmatrix} 0 & -K \\ -K & D \end{pmatrix} \begin{pmatrix} x \\ \lambda x \end{pmatrix}.$$

There are computational difficulties with both the approaches. For details, see **Section 16.3**. *The state-of-the-art numerical techniques are capable of computing only a few eigenvalues and eigenvectors of $P(\lambda)$, especially, if the matrices M , K , and D are large and sparse. The Jacobi-Davidson method for quadratic pencils is one such technique* This method has been briefly described in Section 16.3.

II. Controllability, Observability, Stability and Stabilizability

Basic concepts of stability, stabilizability, controllability and observability, etc., are introduced and a few theoretical results are stated and proved in **Section 16.5**.

In particular, an elementary proof of the **classical Rayleigh stability criterion (Theorem 16.5.4)** is given here using a theoretical result (**Theorem 16.5.3**) on the real and imaginary parts of the eigenvalues of $P(\lambda)$.

III. First-order and Modal Methods for Vibration Control

A natural way to stabilize a vibrating system or to combat dangerous vibrations, such as resonance, is to apply feedback control forces to the structure by using control of the form Bu , yielding the control system:

$$M\ddot{q}(t) + D\dot{q}(t) + Kq(t) = Bu(t).$$

There are two standard approaches for a vibration control problem in the literature. The first approach is to **transfer the second-order system to a standard first-order state-space form** and then apply well-established techniques for the problem in that form.

The second approach is the **Independent Modal Space Control (IMSC)** approach. These two approaches and their computational and engineering difficulties are described in **Section 16.4**.

Solution via First-order State-space Form

The above control system can be written either in the standard form:

$$\dot{x} = Ax + \hat{B}u$$

where

$$A = \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix}, \quad \hat{B} = \begin{pmatrix} 0 \\ M^{-1}B \end{pmatrix}$$

or in the descriptor form:

$$E\dot{z} = Az + \hat{B}u.$$

There are several choices for A and E . These expressions for A and E are given by (16.2.4)-(16.2.7). The matrix $\hat{B} = \begin{pmatrix} 0 \\ B \end{pmatrix}$.

The **numerical difficulties** with standard first-order representation include **possible ill-conditioning of the mass matrix M** and **loss of sparsity, definiteness and bandness of the matrices M, D , and K** , which are very often assets for solving large-scale practical problems.

Numerical methods for the descriptor control systems are not still very well-developed and thus, there are not many opportunities to take advantage of, once the second-order control system is reduced to a descriptor control system. Moreover, in case E is nonsingular, possible ill-conditioning of this matrix, will be a concern.

Independent Modal Space Control (IMSC) Approach

The basic idea behind IMSC approach is to decouple the control system into n independent control systems.

For decoupling of the open-loop system, the knowledge of the entire spectrum of the pencil $P(\lambda)$ and the corresponding eigenvectors is required, which, as stated earlier, can not be obtained in a numerically effective way using the state-of-the-art techniques, if the problem is large. Theoretically, decoupling of M, D and K is possible if **modal damping** is assumed; that is, if the following commutativity relation holds:

$$DM^{-1}K = KM^{-1}D.$$

Such an assumption is, however, hardly satisfied in practice. Even with assumption on modal damping, for decoupling of the *modal control system*, “*very stringent requirements on the location and number of sensors and actuators*” (Inman (1989), pp. 173) are to be satisfied.

IV. Direct and Partial-Modal Approach for Feedback Problems

In view of the above discussions on numerical and engineering difficulties with “**first-order reduction**” and **IMSC** approaches for second-order control problems, it is natural to investigate if numerical algorithms can be developed that require only a minimal amount of spectral information and do not rely on reduction to a first-order form.

Such algorithms, which are called “**direct and partial-modal**” algorithms, for certain state-feedback control problems have been developed in the last few years and an active research on this topic is underway. The algorithms are called “**direct**”, because they do not rely on a first-order transformation and work exclusively in second-order setting. They are “**partial-modal**”, because only a partial knowledge of eigenvalues and eigenvectors of the pencil $P(\lambda)$ is required. The problems that have been solved using “*direct and partial-modal*” approach include *feedback stabilization, partial eigenvalue and eigenstructure assignment and robust eigenvalue assignment* problems.

The *feedback stabilization* problem is the problem of modifying stiffness and damping matrices using feedback such that the modified system is asymptotically stable.

The *partial-eigenvalue assignment* problem is the problem of reassigning of few troublesome eigenvalues of the system to suitably chosen ones by the design engineers, leaving the rest of the spectrum unchanged.

The *partial eigenstructure assignment* problem concerns with assigning not only a few desired eigenvalues, but also a set of eigenvectors as well.

The *robust eigenvalue assignment* problem discussed here, is the problem of assigning the complete set of eigenvalues of the second-order system in such a way that the closed-loop eigenvectors are as well-conditioned as possible.

Mathematical definitions of these problems and descriptions of “direct and partial modal” solutions are given in **Section 16.7**.

Algorithm 16.7.1 is a non-modal **stabilizing algorithm**. No knowledge of eigenvalues and eigenvectors is required to implement this algorithm.

Algorithm 16.7.2 and **Algorithms 16.7.3** are partial-modal algorithms, respectively, for **single-input and multi-input partial eigenvalue assignment problems**. These algorithms can be implemented just by knowing the few eigenvalues (and the corresponding eigenvectors) of the pencil $P(\lambda)$ that are to be reassigned.

Algorithm 16.7.4 is a partial-modal algorithm for **partial eigenstructure assignment problem**. This algorithm also requires the knowledge of only a few eigenvalues (and the corresponding eigenvectors) of $P(\lambda)$ that are to be reassigned. However, since the eigenstructure assignment may not be solvable if the control matrix B is given a priori, the algorithm not only computes the feedback matrices, but computes the control matrix B as well.

Algorithm 16.7.5 and **Algorithm 16.7.6** are **robust eigenvalue assignment algorithms**. These algorithms are direct and compute feedback matrices to modify the mass and stiffness matrices such that the modified system not only has a desired set of eigenvalues but the eigenvectors of the closed-loop system are such that they are as well-conditioned as possible. *Because of this direct and partial-modal nature, these algorithms are very suitable for practical applications.*

V. Orthogonality and Rayleigh-Quotient, and Eigenvalue Bounds of the Quadratic Matrix Pencil.

Development of nonmodal and partial modal algorithms mentioned in the preceding sections are based on certain **orthogonality relations** between the eigenvectors of the pencil $P(\lambda) = \lambda^2 M + \lambda D + K$.

These relations and the associated results on Rayleigh quotient are stated and proved in **Section 16.6**. These new results (**Theorem 16.6.5** and **Theorem 16.6.8**) generalize the known results in the literature on these topics for symmetric matrices and symmetric definite linear pencils. They are, besides their role in algorithm development, of independent interests and useful contributions to linear algebra and vibration engineering literatures. This section also contains

two **computable bounds** (**Theorem 16.6.1** and **Theorem 16.6.2**) for eigenvalues of the pencil $P(\lambda)$. These bounds are computable in the sense that they can be computed just by knowing the largest and smallest real parts of the data matrices M, D , and K , for which numerically viable techniques exist, even for large and sparse problems.

16.9 Chapter Notes and Further Reading

The authoritative books by Gohberg, Lancaster and Rodman (1982, 1983) and the classical book by Lancaster (1966) are valuable sources of knowledge on theory of matrix polynomials. For computational aspects of quadratic matrix polynomials, see Fokkema et al. (1998), Guo et al. (1995), Parlett and Chen (1990), Sleijpen et al. (1996a, 1996b, 1999). Descriptions of algorithms for quadratic matrix eigenvalue problems and associated software can also be found in Bai, et al. (2000). A delightful recent survey on the quadratic matrix eigenvalue problem by Tisseur and Meerbergen (2001) is certainly worth reading. See also Meerbergen (2001). For backward error and conditioning of the quadratic eigenvalue problem, see Tisseur (2000). An alternative recent tool for analyzing the system behavior such as the stability, resonance, etc., is “pseudospectra”. See the survey paper by Trefethen (1999) in this context.

The notion of pseudospectra for a single matrix has been recently extended to matrix polynomials and their uses in control theory (such as finding the distance to the nearest non-regular matrix polynomial and the distance to the nearest uncontrollable system, and others) have been illustrated in Tisseur and Higham (2001) and Higham and Tisseur (2001b).

The books and papers on inverse eigenvalue problems for linear and quadratic matrix pencils, include Gladwell (1986), Inman and Kress (1995), Inman (1989), Ram and Braun (1990, 1993), Ram and Caldwell (1992), Ram and Gladwell (1994), Ram, Blech and Braun (1990), Starek and Inman (1995), Lancaster and Maroulas (1987). For applications giving rise to matrix second-order control systems, the literature is abundant; see, e.g., Balas (1982), Bhaya and Desoer (1985), Joshi (1989) and Soong (1990), for applications to large flexible space structures.

For material on stability of matrix second-order systems, see Barkwell and Lancaster (1992), Barkwell, Lancaster and Markus (1992), Duffin (1960), Hughes and Gardner (1975), Inman (1989), Walker and Schmitendorf (1973), Wimmer (1974), Datta (1999), Zajac (1964).

For concepts and results on controllability, observability, detectability, etc., of second-order systems, see Hughes and Skelton (1980), Laub and Arnold (1984).

Material of Section 16.4 on modal control can be found in the books by Inman (1989), and Mierovitch (1990). See also Andry et al. (1986), Balas (1982), Bhaya and Desoer (1985), Joshi (1989), Soong (1990).

Material of Section 16.5 on nonmodal and partial-modal approaches for feedback stabilization and partial eigenvalue and eigenstructure assignment have been taken from the recent work of Datta and Rincón (1993), Chu and Datta (1996), Datta, Elhay and Ram (1997), Datta and Sarkissian (2001), Datta, Elhay, Ram and Sarkissian (2000) and Sarkissian (2001).

Specifically, the theoretical expressions (**Theorem 16.5.4**) on the real and imaginary parts of the eigenvalues, and the computable bounds (**Theorem 16.6.1** and **Theorem 16.6.2**), of the eigenvalues of a quadratic pencil have been taken from Datta and Rincón (1993). For results on bounds on responses, see Inman (1989), Yae and Inman (1987), etc. See also the recent work of Higham and Tisseur (2001a).

The new results on the orthogonality of eigenvectors (**Theorem 16.6.5** and **Theorem 16.6.8**) on the Rayleigh quotient of symmetric quadratic matrix pencil have been taken from Datta,

Elhay, and Ram (1997).

The nonmodal stabilizing algorithm (**Algorithm 16.7.1**), the algorithm for the single-input partial eigenvalue assignment problem (**Algorithm 16.7.2**), the algorithm for the multi-input partial eigenvalue assignment problem (**Algorithm 16.7.3**), and the algorithms for the robust eigenvalue assignment (**Algorithm 16.7.5** and **Algorithm 16.7.6**) appear, respectively, in Datta and Rincón (1993), Datta, Elhay and Ram (1997), Datta and Sarkissian (2001)), and Chu and Datta (1996). An algorithm for a variation of the partial eigenvalue assignment that assigns a given set of eigenvalues and guarantees only the stability of the remaining eigenvalues, but not their invariance, appears in Datta, Elhay and Ram (1996). This algorithm has not been included in this Chapter.

Also, the problem of output feedback has not been considered here. See Inman (1989) and Inman and Kress (1995) for some results on this problem.

Finally, it is worth mentioning that the partial eigenstructure assignment problem and the **eigenvalue embedding problem** (recently discussed in the literature) are closely related to a problem of immense practical importance in vibration studies called **Finite-Element Model Updating** (FEMU) problem: this problem is routinely solved in aerospace, automobile and other structural industries.

The FEMU problem is the problem of updating a symmetric finite element generated matrix second-order model such that the updated model is symmetric and a set of measured frequencies and mode shapes from a realized practical structure are reproduced in this updated model, while the remaining eigenvalues and eigenvectors of the FEM remain unchanged or stay within the frequency range of interests after updating. Thus, **the FEMU problem is a constrained eigenstructure assignment problem**; the constraint is that the updated model has to remain symmetric.

The eigenvalue embedding problem concerns with embedding a given set of eigenvalues in a symmetric second-order model such that the updated model remain symmetric. Then, it solves the FEMU problem except that the measured mode shapes are not assigned. The eigenvalue embedding problem has been studied recently in Carvalho, Datta, Lin and Wang (2001), and Ferng, Lin, Pierce and Wang (2001).

For learning more on the FEMU problem, see the authoritative book by Friswell and Mottershead (1995) and the discussions in the recent survey paper by Datta (2002). For solutions of the FEMU problem using eigenstructure assignment, see Inman and Minas (1990) and Zimmerman and Widengren (1990).

The algorithm for the partial eigenstructure assignment (**Algorithm 16.7.4**) appears in Datta, Elhay, Ram and Sarkissian (2000). Several other interesting results on partial eigenvalue and eigenstructure assignment problems, both for first-order and second-order systems are contained in the recent Ph.D dissertation of Sarkissian (2001). Although discussions on feedback control in distributed parameter systems has not been included in this Chapter, there have been some recent work, generalizing algorithms for partial eigenvalue assignment in second-order systems (**Algorithms 16.7.2** and **16.7.3**) to the distributed parameter models. These algorithms,

like their matrix second-order counter parts, are also “**direct and partial-modal**” in the sense that these algorithms work exclusively with the distributed parameter models without requiring discretization to matrix second-order systems, and require only a small finite set of the infinite open-loop spectrum (only those which need to be reassigned) and associated eigenvectors for implementations. For details of these methods, see Ph.D. dissertation of Sarkissian (2001), and Datta, Ram and Sarkissian (2001), and Datta and Sarkissian (2002).

EXERCISES

1. (Modal Controllability Criterion (Hughes and Skelton (1980)))

- (a) Let $M = M^T > 0$, and $K = K^T$. Then prove that the undamped system

$$M\ddot{q}(t) + Kq(t) = Bu(t),$$

is controllable if and only if the $n \times 2n$ matrix $R = (\tilde{B}, \Lambda_k \tilde{B}, \dots, \Lambda_k^{n-1} \tilde{B})$, has full rank, where $\tilde{B} = S_m^T B$ and $\Lambda_k = S_m^T K S_m$; S_m being the modal matrix of the system.

- (b) Prove that if the diagonal entries of Λ_k are distinct, then the system is controllable if and only if

$$\|b_i\| > 0, \quad i = 1, \dots, n,$$

where b_i is the i^{th} row of \tilde{B} .

2. Prove that the second-order system

$$M\ddot{x} + D\dot{x} + Kx = Bu, \quad y = Px + Q\dot{x}$$

is observable if and only if

$$\text{rank} \begin{pmatrix} \lambda Q + P \\ \lambda^2 M + \lambda D + K \end{pmatrix} = n$$

for all eigenvalues λ of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$.

3. (Frequency Response Computation under Modal Damping)

Show that the frequency response matrix of the second-order system in Exercise 2 is given by

$$G(\omega) = (j\omega Q + P)(K + j\omega D - \omega^2 M)^{-1} B,$$

How can this matrix be computed using modal damping?

4. (Model Identification Under Modal Damping)

Assuming modal damping and using simultaneous diagonalization, prove that

$$\begin{aligned} M &= S_m^{-1} (S_m^T)^{-1} \\ D &= S_m^{-1} \text{diag}(2\zeta_1 \omega_1, \dots, 2\zeta_n \omega_n) (S_m^T)^{-1} \\ K &= S_m^{-1} \text{diag}(\omega_1^2, \dots, \omega_n^2) (S_m^T)^{-1} \end{aligned}$$

where S_m is the modal transforming matrix, $\omega_1, \dots, \omega_n$ are natural frequencies and ζ_1, \dots, ζ_n are modal damping ratios.

5. Prove that simultaneous orthogonal triangularization of the matrices M, D , and K is not possible in case of general damping (consult Williams and Laub (1992)).

6. State and prove a theorem analogous to Theorem 16.7.2 in the multi-input case and then use this theorem to derive Algorithm 16.7.3.
7. Using the eigenvalue criterion of controllability (Theorem 16.5.1), prove Theorem 16.7.1.
8. Write the $n \times n$ undamped second-order model $M\ddot{x} + Kx = f$ in the partitioned form:

$$\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

Develop an algorithm for model reduction of the above system in terms of the matrices M_{ij} and K_{ij} , that is, find a matrix P such that $P^T M P$ and $P^T K P$ are of dimensions $\frac{n}{2}$ (consult Inman (1989), pp. 135-137). (**Hint:** Take $P = \begin{pmatrix} I \\ -K_{22}^{-1} K_{21} \end{pmatrix}$).

9. Give a proof of Algorithm 16.7.5.

Research Problems on Chapter 16

1. Sharpen the eigenvalue bounds of the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ in Theorems 16.6.1 and 16.6.2.
2. Derive a computationally efficient algorithm by applying the results on the Rayleigh-quotient for the quadratic pencil $P(\lambda) = \lambda^2 M + \lambda D + K$ in Theorem 16.6.8 or otherwise to compute the frequency response matrix in Exercise 3.
3. Investigate how the freedom offered by Algorithm 16.7.3 can be exploited to turn the algorithm into a robust partial eigenvalue assignment algorithm.

(Note that Algorithms 16.7.5 and 16.7.6 are the robust eigenvalue assignment algorithms for the complete spectrum assignment).

4. Develop an algorithm for model identification of a second-order control system directly in matrix second-order setting; that is, the algorithm should not involve transformation to a first-order system.

References

1. A. N. Andry, E. Y. Shapiro, and K. M. Sobel, *Modal Control and Vibrations, Frequency Domain and State Space Methods for Linear Systems*, pp. 185-198, 1986, (C. Byrnes, and A. Lindquist, Editors)
2. Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. Van der Vorst, *Templates for the solution of Algebraic Eigenvalue Problems - A Practical Guide*, SIAM, Philadelphia, 2000.
3. M.J. Balas, Trends in large space structure control theory: fondest hopes, wildest dreams, *IEEE Trans. Automat. Control*, AC-27, pp. 522-535, 1982.
4. L. Barkwell, and P. Lancaster, *Overdamped and gyroscopic vibrating systems*, *Trans. ASME*, vol. 59, pp. 176-181, 1992.
5. L. Barkwell, P. Lancaster, and A. Markus, *Gyroscopically stabilized systems: a class of quadratic eigenvalue problems with real spectrum*, *Canadian J. Math.*, 44, pp. 42-53, 1992.
6. A. Bhaya and C. Desoer, On the design of large flexible space structures (LFSS), *IEEE Trans. Automat. Control*, AC-30, pp. 1118-1120, 1985.
7. D. Calvetti, B. Lewis and L. Reichel, On the solution of the single-input pole placement problem, *Proc. Mathematical Theory of Networks and Systems*, (MTNS'98), II poliografo, Padova, pp. 585-588, 1998, (A. Beghi, L. Finesso and G. Picci, Editors).
8. D. Calvetti, B. Lewis and L. Reichel, On the selection of poles in the single input pole placement problem, *Lin. Alg. Appl.*, vol. 302/303, pp. 331-345, 1999.
9. J. Carvalho, B.N. Datta, W.-W. Lin, and C.S. Wang, Eigenvalue embedding in a quadratic pencil using symmetric low rank updates, *Fourth SIAM Conference on Linear Algebra in Signals, Systems, and Control*, Boston, MA, August, 2001.
10. E.K. Chu, Approximate pole assignment, *Int. J. Control*, vol. 59, pp. 471-484, 1994.
11. E. K. Chu and B. N. Datta, Numerically robust pole assignment for second-order systems, *Int. J. Control*, vol. 64, pp. 1113-1127, 1996.
12. M.T. Chu, Inverse Eigenvalue Problems, *SIAM Review*, vol. 40, no. 1 pp. 1-39, 1998.
13. B.N. Datta, Finite element model updating, eigenstructure assignment and eigenvalue embedding techniques for vibrating systems, special issue on “*Vibration Control*” of *Mechanical System and Signal Processing*, vol. 16, no. 1, pp. 83-96, 2001.
14. B.N. Datta, Linear and Numerical Linear Algebra in Control Theory: Some Research Problems, *Lin. Alg. Appl.*, vol. 197/198, pp. 755-790, 1984.

15. B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, Pacific Grove, CA, 1995.
16. B.N. Datta, Stability and Inertia, *Lin. Alg. Appl.*, vol. 302-303, pp. 563-600, 1999.
17. B.N. Datta, S. Elhay, and Y. Ram, An Algorithm for the partial multi-input pole assignment of a second-order control system, *Proc. IEEE. Conf. Dec. Control.* pp. 2025-2029, 1996.
18. B.N. Datta, S. Elhay, and Y. Ram, Orthogonality and partial pole assignment for the symmetric definite quadratic pencil, *Lin. Alg. Appl.*, vol. 257, pp. 29-48, 1997.
19. B.N. Datta, S. Elhay, Y.M. Ram and D.R. Sarkissian, Partial eigenstructure assignment for the quadratic pencil, *Journal of Sound and Vibration*, vol. 230, no. 1, pp. 101-110, 2000.
20. B.N. Datta, Y. Ram and D. Sarkissian, Spectral modification for gyroscopic systems, *Z. Angew Math. Mecanics*, vol. 82, pp. 191-200, 2001.
21. B.N. Datta and F. Rincón, Feedback stabilization of a second-order system: a nonmodal approach, *Lin. Alg. Appl.*, vol. 188/189, pp. 135-161, 1993.
22. B.N. Datta and Y. Saad, Arnoldi methods for large Sylvester-like observer matrix equation and an associated algorithm for partial spectrum assignment, *Lin. Alg. Appl.*, vol. 154/155/156, pp. 225-244, 1991.
23. B.N. Datta and D. Sarkissian, Multi-input partial eigenvalue assignment for the symmetric quadratic pencil, *Proc. Amer. Control Conference*, pp. 2244-2247, 1999.
24. B.N. Datta and D. Sarkissian, Theory and computations of some inverse eigenvalue problems for the quadratic pencil, *Contemporary Mathematics*, Amer. Math. Soc. Providence, RI, vol. 20, pp. 221-240, 2001.
25. B.N. Datta and D. Sarkissian, Feedback control in distributed parameter gyroscopic systems: A solution of the partial eigenvalue assignment problem, special issue on “*Vibration Control*” of *Mechanical Systems and Signal Processing*, vol. 16, no. 1, pp. 3-17, 2001.
26. R.J. Duffin, The Rayleigh-Ritz method for dissipative or gyroscopic systems, *Quart. Appl. Math.*, vol. 18, pp. 215-221, 1960.
27. W.R. Ferng, W.-W. Lin, D. Pierce and C.S. Wang, Nonequivalence transformation of λ -matrix eigenproblems and model embedding approach to model tuning, *Num. Lin. Alg. Appl.*, vol. 8, pp. 53-70, 2001.
28. M.J. Friswell and J.E. Mottershead, *Finite Element Modal Updating in Structural Dynamics*, Kluwer Academic Publishers, London, 1995.

29. D.R. Fokkema, G.L.G. Sleijpen and H.A. van der Vorst, Jacobi-Davidson style QR and QZ algorithms for partial reduction of matrix pencils, *SIAM J. Sci. Comput.* vol. 20, no. 1, pp. 94-125, 1998.
30. G.M. Gladwell, *Inverse Problems in Vibration*, Martinus Nijhoff, Dordrecht, the Netherlands, 1986.
31. I. Gohberg, P. Lancaster and L. Rodman, *Matrix Polynomials*, Academic Press, New York, 1982.
32. I. Gohberg, P. Lancaster, and L. Rodman, *Matrices and Indefinite Scalar Products*, Birkhäuser Verlag, Basel, 1983.
33. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Edition, Johns Hopkins University, Baltimore, MD, 1996.
34. R.G. Grimes, J.G. Lewis, and H.D. Simon, A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems, *SIAM J. Matrix Anal. Appl.*, vol. 15, pp. 228-272, 1994.
35. J.S. Guo, W-W. Lin, C-S. Wang, Numerical Solutions of large sparse quadratic eigenvalue problems, *Lin Alg. Appl.*, vol. 225, pp. 57-89, 1995.
36. N.J. Higham and F. Tisseur, Bounds for eigenvalues of matrix polynomials, Numerical Analysis Report 371, *Manchester Centre for Computational Mathematics*, Manchester, England, January 2001a. To appear in *Lin. Alg. Appl.*
37. N.J. Higham and F. Tisseur, *More on pseudospectra for polynomial eigenvalue problems and applications in control theory*, Numerical Analysis Report 372, Manchester Centre for Computational Mathematics, Manchester, England, January 2001b, to appear in *Lin. Alg. Appl.*
38. N.J. Higham, F. Tisseur, and P.M. Van Dooren, Detecting a definite Hermitian pair and a hyperbolic or elliptic quadratic eigenvalue problem, and associated nearness problems, to appear in *Lin. Alg. Appl.*
39. P.C. Hughes, and L.T. Gardner, Asymptotic stability of linear stationary mechanical systems, *ASME J. Applied Mechanics*, vol. 42, pp. 228-229, 1975.
40. P.C. Hughes and R.E. Skelton, Controllability and observability of linear matrix-second-order systems, *J. Appl. Mechanics*, vol. 47, pp. 415-420, 1980.
41. D.J. Inman, *Vibration with Control, Measurement and Stability*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
42. D.J. Inman and A. Kress, Eigenstructure assignment via inverse eigenvalue methods, *AIAA J. Guidance, Control and Dynamics*, vol. 18, pp. 625-627, 1995.

43. S.M. Joshi, *Control of Large Flexible Space Structures, Lecture Notes in Control and Inform. Sci.*, vol. 131, Springer-Verlag, Berlin, 1989.
44. J. Juang and P.G. Maghami, Robust eigensystem assignment for second-order dynamics systems, *Mechanics and Control of Large Flexible Structures*, Progress in Astronautics and Aeronautics, AIAA, Washington DC, vol. 129, pp. 373–388, 1990.
45. J. Juang and P.G. Maghami, Robust eigensystem assignment for state estimators using second-order models, *J. Guidance, Control and Dynamics*, vol. 15, pp. 920–927, 1992.
46. J. Kautsky, N.K. Nichols and P. Van Dooren, Robust pole assignment in linear state feedback, *Int. J. Control*, vol. 41, no. 5, pp. 1129–1155, 1985.
47. L. Komzsik, *MSC/NASTRAN Numerical Methods User's Guide*, Version 70.5, The MacNeal-Schwendler Corporation, Los Angeles, 1998.
48. P. Lancaster, Private Communication (1997).
49. P. Lancaster, *Lambda-Matrices and Vibrating Systems*, Pergamon Press, Oxford, UK, 1966.
50. P. Lancaster, and J. Maroulas, Inverse eigenvalue problems for damped vibrating systems, *J. Math. Anal. Appl.*, vol. 123, pp. 238-261, 1987.
51. P. Lancaster, and M. Tismenetsky, Inertia Characterization of self-adjoint matrix polynomials, *Lin. Alg. Appl.*, pp. 479-496, 1983.
52. P. Lancaster and M. Tismenetsky, *The Theory of Matrices with Applications*, 2nd Edition, Academic Press, New York, 1985.
53. A. J. Laub, and W. F. Arnold, Controllability and observability criteria for multivariate linear second order models, *IEEE Trans. Auto. Control*, vol. AC-29, pp. 163-165, 1984.
54. K. Meerbergen, Locking and restarting quadratic eigenvalue solvers, *SIAM J. Sci. Comput.*, vol. 22, pp. 1814-1839, 2001.
55. L. Meirovitch, *Dynamics and Control of Structures*, John Wiley & Sons, New York, 1990.
56. C. Minas and D.J. Inman, Matching finite element models to modal data, *Trans. ASME*, vol. 112, pp. 84-92, 1990.
57. B.N. Parlett, *The Symmetric Eigenvalue Problem*, Englewood Cliffs, NJ, Prentice Hall, 1980.
58. B.N. Parlett and H.C. Chen, Use of indefinite pencil for computing damped natural modes, *Lin. Alg. Appl.*, vol. 140, pp. 53-88, 1990.

59. B.N. Parlett and Y. Saad, Complex shift and invert strategies for real matrices, *Lin. Alg. Appl.*, vol. 88-89, pp. 575-595, 1987.
60. Y.M. Ram, An inverse eigenvalue problem for a modified vibrating system, *SIAM J. Appl. Math.*, vol. 53, pp. 1762-1775, 1993.
61. Y.M. Ram, J.J. Blech, and S.G. Braun, Eigenproblem error bounds with application to the symmetric dynamic system modification, *SIAM J. Matrix Anal. Appl.*, vol. 11, no. 4, pp. 553-564, 1990.
62. Y.M. Ram and S.G. Braun, Upper and lower bounds for the natural frequencies of modified structures based on truncated modal testing results, *Journal of Sound and Vibration*, vol. 137, no. 1, pp. 69-81, 1990.
63. Y.M. Ram and S.G. Braun, An inverse problem associated with the dynamic modification of structures, *ASME J. Appl. Mechanics*, vol. 58, pp. 233-237, 1991.
64. Y.M. Ram and J. Caldwell, Physical parameters reconstruction of a free-free mass-spring system from its spectra, *SIAM J. Appl. Math.*, vol. 52, pp. 140-152, 1992.
65. Y.M. Ram and G.M.L. Gladwell, Constructing a finite element model of a vibratory rod from eigendata, *Journal of Sound and Vibration*, vol. 169, pp. 229-237, 1994.
66. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halstead Press, New York, 1992.
67. Y. Saad, *Iterative Methods for Sparse Linear Systems*, Boston: PWS Publishing, Boston, MA, 1996.
68. Y. Saad, Numerical solution of large nonsymmetric eigenvalue problems, *Comp. Phys. Comm.*, vol. 53, pp. 71-90, 1989.
69. D. Sarkissian, *Theory and Computations of Partial Eigenvalue and Eigenstructure Assignment Problems in Matrix Second-order and Distributed Parameter Systems*, Ph.D. Dissertation, Northern Illinois University, DeKalb, Illinois, 2001.
70. G.L.G. Sleijpen, G.L. Booten, D.R. Fokkema, and H.A. van der Vorst, Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems, *BIT*, vol. 36, no. 3, pp. 595-633, 1996a.
71. G.L.G. Sleijpen, H.A. van der Vorst and M. van Gijzen, Quadratic eigenproblems are no problem, *SIAM News*, vol. 29 pp. 8-9, 1996b.
72. G.L.G. Sleijpen, H.A. van der Vorst and Z. Bai, Jacobi-Davidson algorithms for various eigenproblems - A working document, *H.A. van der Vorst's homepage* (1999).

73. T.T. Soong, *Active Structural Control: Theory and Practice*, Longman Scientific Technical, Essex, UK, 1990.
74. L. Starek and D.J. Inman, A symmetric inverse vibration problem with overdamped modes, *J. Sound and Vibration*, vol. 181, pp. 893-903, 1995.
75. G.W. Stewart and Ji-guang Sun, *Matrix Perturbation Theory*, Academic Press, San Diego, CA, 1990.
76. F. Tisseur, Backward error and condition of polynomial eigenvalue problems, *Lin. Alg. Appl.*, vol. 309, pp. 339-361, 2000.
77. F. Tisseur and N.J. Higham, Structured pseudospectra for polynomial eigenvalue problems, with applications, *SIAM J. Matrix Anal. Appl.*, vol. 23, pp. 187-208, 2001.
78. F. Tisseur and K. Meerbergen, The quadratic eigenvalue problem, *SIAM Review*, vol. 43, no. 2, pp. 235-286, 2001.
79. L.N. Trefethen, Computation of pseudospectra, *Acta Numerica*, vol. 8, pp. 247-295, 1999.
80. K. Veselić, A Jacobi eigenreduction algorithm for definite matrix pairs, *Numer. Math.*, vol. 64, pp. 241-268, 1993.
81. J.A. Walker and W.E. Schmitendorf, A simple test for asymptotic stability in partially dissipative symmetric systems, *ASME J. Appl. Mechanics*, vol. 40, pp. 1120-1121, 1973.
82. T. Williams and A.J. Laub, Orthogonal canonical forms for second-order systems, *IEEE Trans. Automat. Control*, vol. 37, pp. 1050-1052, 1992.
83. Wimmer, H., Inertia theorems for matrices, controllability, and linear vibration, *Lin. Alg. Appl.*, vol. 8, pp. 457-461, 1974.
84. K.H. Yae and D.J. Inman, Response bounds for linear underdamped systems, *ASME J. Appl. Mechanics*, vol. 54, no. 2, pp. 419-423, 1987.
85. E.E. Zajac, The Kevin-Taft-Chetaev Theorem and extensions, *J. Astro. Sc.*, vol. 11, pp. 46-49, 1964.
86. D.C. Zimmerman and M. Windengren, Correcting finite element model using a symmetric eigenstructure assignment technique, *AIAA*, vol. 28, pp. 1670-1676, 1990.

Appendix A

SOME EXISTING SOFTWARE FOR CONTROL SYSTEMS DESIGN AND ANALYSIS

In this Chapter, we will give a brief description of some of the existing software for linear time-invariant control system design and analysis.

A.1 MATLAB CONTROL SYSTEM TOOLBOX

As the title suggests, MATLAB *Control System Toolbox* is based on the well-known matrix computations software “MATLAB”. It is a collection of M-files which implement some of the numerically viable algorithms for control system design, analysis, and modeling.

The control systems can be modeled either as transfer functions or in state space form. Both continuous-time and discrete-time systems can be handled. The toolbox has excellent graphic capabilities and then various time and frequency responses can be viewed on the screen and analyzed.

The software can be obtained from

The MathWorks, Inc.

24 Prime Park Way

Natick, MA 01760-1500

Tel: (508) 647-7000

Fax: (508) 647-7001

URL: <http://www.mathworks.com>

Newsgroup: Comp. soft. sys. matlab.

See MATLAB Control System Toolbox: Users Guide (1996) for details.

A.2 MATCONTROL

MATCONTROL is also a collection of **M-files implementing major algorithms of this book**. MATCONTROL is primarily designed for class-room use – by using this toolbox, the students (and the instructors) will be able to compare different algorithms for the same problem with respect to efficiency, stability, accuracy, easiness-to-use and specific design and analysis requirements.

A.3 CONTROL SYSTEM PROFESSIONAL – ADVANCED NUMERICAL METHODS (CSP-ANM).

Control System Professional (CSP) based on “Mathematica” is a collection of *Mathematica* programs to solve control system problems. *Control System Professional–Advanced Numerical Methods (CSP-ANM)* extends the scope of CSP by adding new numerical methods for a wide class of control problems as well as for a number of matrix computations problems that have an extensive use in control systems design and analysis.

Advanced Numerical Methods is compatible with, and requires, *Control System Professional* 2.0 or later. The software has been developed by Biswa Nath Datta and Daniil Sarkissian (with the help of Igor Bakshee from Wolfram Research Incorporation).

“Typically, *Advanced Numerical Methods* provides several numerical methods to solve each problem enabling the user to choose from most appropriate tool for a particular task based on computational efficiency and accuracy.” Thus, the package, though oriented mostly for professional users, is also an important tool for students, researchers, and educators alike.

The algorithms implemented in the package have been taken mostly form the current book by the author.

Software and Manual: There is a User’s Manual written by Biswa Nath Datta and Daniil Sarkissian with help from Igor Bakshee and published by *Wolfram Research, Inc.*. Both the software and the manual can be obtained from:

Wolfram Research, Inc.
100 Trade Center Drive
Champaign, Illinois 61820-7237
USA
phone: (217) 398-0700
Fax: (217) 398-0747
E-mail: Info@wolfram.com
URL: www.wolfram.com

A.4 SLICOT

SILCOT is a Fortran 77 Subroutine Library in Control Theory. It is built on the well-established matrix software packages, the **B**asic **L**inear **A**lgebra **P**ackage (BLAS) and the **L**inear **A**lgebra **P**ackage (LAPACK). The library also contains other mathematical tools such as discrete sine/cosine and Fourier transformations. The routines can be embedded in MATLAB by an appropriate interface thus enhancing the applicability of the library.

For a brief description of the library, see the paper “**SLICOT—A Subroutine Library in Systems and Control Theory**” by Peter Benner, Volker Mehrmann, Vasile Sima, Sabine Van Huffel, and Andras Varga in *Applied and Computational Control, Signals, and Circuits* (Biswa Nath Datta, Editor), Birkhauser, 2001. Currently, the user manual is available as on-line documentation in HTML format (filename libindex.html) at:

<http://www.win.tue.nl/wgs/slicot.html>.

A.5 MATRIX_X

MATRIX_X, as the title suggests, is built on functions that are most commonly used for matrix computations. It is broken into several modules. The principal ones are *MATRIX_X* Core, Control, and System Build, Optimization and Robust Control. The core module contains

the core MATLAB commands with some modifications and extensions. The control module contains both classical and modern control commands.

The *MATRIX_X* core and control modules are command driven, while the system build module is menu driven. This module allows the users to simulate the systems by building the block diagrams of the systems on the screen. *MATRIX_X* is a product of *Integrated Systems, Inc.* There exist a *MATRIX_X* user's guide (1991) and a book by Shahian and Hassul (1992) describing the functional details of the software.

A.6 SYSTEM IDENTIFICATION SOFTWARE

Each of the software packages **MATLAB Control System Toolbox**, **Control System Professional**, **Control System Professions–Advanced Numerical Methods**, **SLICOT**, **MATRIX_X**, etc., has its own software module for system identification. See Chapter 9 of this book for details.

There now also exist a few software packages, especially designed for system identification. We describe three of them in the following.

A.6.1 MATLAB System Identification Toolbox

This toolbox has been developed by Prof. Lennart Ljung of Linköping University, Sweden. The toolbox can be used either in command mode or via a Graphical User Interface (GUI). The details can be found in the Users' manual (Ljung (1991)) and MathWorks website:

<http://www.mathworks.com>

A.6.2 Xmath Interactive System Identification Module, Part-2

This is a product of Integrated System Inc., Santa Clara, USA, 1994. It is a GUI-based software for multivariable system identification. The details can be found in User's Manual (Overschee, DeMoor, Aling, Kosut and Boyd (1994)).

Website: http://www.isi.com/products/MATRIX_X/Techspec/MATRIX_X-Xmath/xm36.html.

A.6.3 ADAPT_X

This software package has been developed by W.E. Larimore. For details, see the Users Manual (Larimore (1997)).

Website: <http://adaptics.com>

Some further details on these softwares and subspace state-space system identification software can be found in the recent paper by DeMoor, Van Overschee and Favoreel (1999).

References

1. *MATLAB Control System Toolbox*: User's Guide, The MathWorks, Inc. Natick, MA, 1996.
2. *MATRIX_X User's Guide*, Integrated Systems, Inc., Santa Clara, CA, 1991.
3. *MATHEMATICA Control System Professional*, Wolfram Research Inc., Champaign, Illinois, 1996.
4. B.N. Datta and D. Sarkissian (with I. Bakshee), *Control System Professional—Advanced Numerical Methods*: User's Guide, Wolfram Research Inc., Champaign, Illinois, 2001.
5. B. DeMoor, P. VanOverschee, and W. Favoreel, Subspace State-Space System Identification, *Applied and Computational Control, Signals, and Circuits*, Birkhauser, Boston, pp. 247-311, (B.N. Datta, et al., Editors), 1999.
6. W.E. Larimore, *ADAPT_X Automatic System Identification Software*, Users Manual, Adap-tics Inc., Reading, MA 01867, USA, 1997.
7. L. Ljung, *System Identification Toolbox for Use with MATLAB*, The MathWorks Inc., MA, USA, 1991.
8. B. Shahian and M. Hassul, *Control System Design Using MATRIX_X* Prentice Hall, Englewood Cliffs, NJ, 1992.
9. P. VanOverschee, B. DeMoor, H. Aling, R. Kosut, S. Boyd, *Xmath Interactive System Identification Module, Part 2*, Integrated Systems, Inc., Santa Clara, CA, 1994.

Appendix B

MATCONTROL AND LISTING OF MATCONTROL FILES

ABOUT MATCONTROL:

What is the MATCONTROL library?

The MATCONTROL library is a set of M-files implementing the majority of algorithms of the book:

Numerical Algorithms for Linear Control Systems Design and Analysis, by B.N. Datta.

Who wrote the MATCONTROL library?

The MATCONTROL library was written by several graduate students of Professor Datta. The most contributions were made by Joao Carvalho and Daniil Sarkissian.

How can I get the MATCONTROL library?

The MATCONTROL library is distributed with the book mentioned above.

What to do if a routine is suspected to give wrong answers?

Please let us know immediately. Send an email to dattab@math.niu.edu and, if possible, include a MATLAB diary file that calls the routine and produces the wrong answer.

How to install MATCONTROL:

The MATCONTROL library is distributed in a subdirector called “Matcontrol”.

This directory must be copied from the media that accompanies the book into anywhere in your system.

After the “Matcontrol” directory has been copied, you just have to let MATLAB know where MATCONTROL is located. In order to do that, you must include it in MATLAB’s path.

The easiest way to do so is by including the proper MATLAB commands in your MATLAB startup file (startup.m). If you do not have this file already, please create it.

Using your preferred text editor, open (or create) startup.m and add the following line:

Unix/Linux systems:

```
matlabpath([matlabpath,'path_of_Matcontrol']);
```

MS-Windows* systems:

```
path(path,'path_of_Matcontrol');
```

Examples: Here, “Mfiles” is the working directory of MATLAB.

On Linux PC:

```
matlabpath([matlabpath,:/home/carvalho/Mfiles/Matcontrol']);
```

On Unix-Solaris Workstation:

```
matlabpath([matlabpath,:/export/home/grad/carvalho/Mfiles/Matcontrol']);
```

On MS-Windows PC:

```
path(path,'C:\Mfiles\Matcontrol');
```

Once you’ve done that, you can use MATCONTROL in the next MATLAB session. Please issue the command **“help Matcontrol”** to see if MATCONTROL was properly included in MATLAB’s path. You should see a list of all MATCONTROL M-files.

*Disclaimer: MATLAB and Windows are trademarks of their respective owners.

CHAPTER-WISE LISTING OF MATCONTROL FILES

Here is the Chapter-wise listing of MATCONTROL files.

Reference: Numerical Algorithms for Linear Control Systems Design and Analysis, by B.N. Datta.

Chapter 5: Linear State Space Models and Solutions of the State Equations

- | | |
|----------|--|
| EXPMPADE | - The Padé approximation to the exponential of a matrix |
| EXPMSCR | - Computing the exponential of a matrix using Schur decomposition |
| EXMPHESS | - Computing the exponential of a matrix using Hessenberg decomposition |
| FREQRESH | - Computing the frequency response matrix using Hessenberg decomposition |
| INTMEXP | - Computing an integral involving a matrix exponentials |

Chapter 6: Controllability, Observability and Distance to Uncontrollability

- | | |
|----------|---|
| CNTRLHS | - Finding the controller-Hessenberg form |
| OBSERHS | - Finding the observer-Hessenberg form |
| CNTRLC | - Find the controller canonical form (Lower Companion) |
| DISCNTRL | - Distance to controllability using the Wicks-DeCarlo algorithm |

Chapter 7: Stability, Inertia and Robust Stability

- | | |
|----------|--|
| INERTIA | - Determining the inertia and stability of a matrix without solving a matrix equation or computing eigenvalues |
| H2NRMCG | - Finding H_2 -norm using the controllability Grammians |
| H2NRMOG | - Finding H_2 -norm using the observability Grammian |
| DISSTABC | - Determining the distance to the continuous-time stability |

DISSTABD	- Determining the distance to the discrete-time stability
ROBstab	- Robust stability analysis using Lyapunov equations
Chapter 8:	Numerical Solutions and Conditioning of Lyapunov and Sylvester Equations
CONDSYLVC	- Finding the condition number of the Sylvester equation problem
LYAPCHLC	- Finding the Cholesky factor of the positive definite solution of the continuous-time Lyapunov equation
LYAPCHLD	- Find the Cholesky factor of the positive definite solution of the discrete-time Lyapunov equation
LYAPCSD	- Solving discrete-time Lyapunov equation using complex-Schur decomposition of A
LYAPFNS	- Solving continuous-time Lyapunov equation via finite series method
LYAPHESS	- Solving continuous-time Lyapunov equation via Hessenberg decomposition
LYAPRSC	- Solving the continuous-time Lyapunov equation via real-Schur decomposition
LYAPRSD	- Solving discrete-time Lyapunov equation via real-Schur decompostion
SEPEST	- Estimating the <i>sep</i> function with triangular matrices
SEPKR	- Computing the <i>sep</i> function using Kronecker product
SYLVHCSC	- Solving the Sylvester equation using Hessenberg and complex Schur decompositions
SYLVHCSD	- Solving the discrete-time Sylvester equation using Hessenberg and complex-Schur decompositions
SYLVHESS	- Solving the Sylvester equation via Hessenberg decomposition
SYLVHRSC	- Solving the Sylvester equation using Hessenberg and real Schur decompositions
SYLVHUTC	- Solving an upper triangular Sylvester equation
Chapter 9:	Realization and Subspace Identification
MINRESVD	- Finding minimal realization using singular value decomposition of Hankel matrix of Markov parameters (Algorithm 9.3.1)
MINREMSVD	- Finding minimal realization using singular value decomposition of Hankel matrix of lower order (Algorithm 9.3.2)
Chapter 10:	Feedback Stabilization, Eigenvalue Assignment, and Optimal Control
STABLYAPC	- Feedback stabilization of continuous-time system using Lyapunov equation
STABLYAPD	- Feedback stabilization of discrete-time system using Lyupunov equation
STABRADC	- Finding the complex stability radius using the bisection method
HINFNRM	- Computing H_∞ -norm using the bisection method
Chapter 11:	Numerical Methods and Conditioning of the EVA Problems
POLERCS	- Single-input pole placement using the recursive algorithm
POLEQRS	- Single-input pole placement using the QR version of the recursive algorithm
POLERQS	- Single-input pole placement using RQ version of the recursive algorithm

- POLERCM - Multi-input pole placement using the recursive algorithm
- POLERCX - Multi-input pole placement using the modified recursive algorithm that avoids complex arithmetic and complex feedback.
- POLEQRM - Multi-input pole placement using the explicit QR algorithm
- POLESCH - Multi-input pole placement using the Schur decomposition
- POLEROB - Robust pole placement

Chapter 12: State Estimation: Observer and Kalman Filter

- SYLVOBSC - Solving the constrained multi-output Sylvester-observer equation
- SYLVOBSM - Solving the multi-output Sylvester-observer equation
- SYLVOBSMB - Block triangular algorithm for the multi-output Sylvester-observer equation

Chapter 13: Numerical Solutions and Conditioning of the Algebraic Riccati Equations

- RICEIGC - The eigenvector method for the continuous-time Riccati equation
- RICSCHC - The Schur method for the continuous-time Riccati equation
- RICSCHD - The Schur method for the discrete-time Riccati equation
- RICGEIGD - The generalized eigenvector method for the discrete-time Riccati equation
- RICNWTNC - Newton's method for the continuous-time Riccati equation
- RICNWTND - Newton's method for the discrete-time Riccati equation
- RICSGNC - The matrix sign-function method for the continuous-time Riccati equation
- RICSGND - The matrix sign-function method for the discrete-time Riccati equation
- RICNWLS - Newton's method with line search for the continuous-time Riccati equation
- RICNWLD - Newton's method with line search for the discrete-time Riccati equation

Chapter 14: Internal Balancing and Model Reduction

- BALSV - Internal balancing using the singular value decomposition
- BALSQT - Internal balancing using the square-root algorithm
- MODREDS - Model reduction using the Schur method
- HNAPRX - Hankel norm approximation