CHAPTER 1

INTRODUCTION AND OVERVIEW

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

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

$$y(t) = Cx(t) + Du(t), \tag{1.0.2}$$

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 $r \times m$ 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:

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

$$y(k) = Cx(k) + Du(k). \tag{1.0.4}$$

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.

The following major topics, associated with the design and analysis of linear control system have been addressed in the book: (i) Linear and Numerical Linear Algebra, (ii) System Responses, (iii) Controllability and Observability, (iv) Stability and Inertia, (v) Lyapunov, Sylvester and Riccati Equations, (vi) Realization and Identification, (vii) Feedback Stabilization and Eigenvalue Assignment, (viii) State Estimation, (ix) Internal Balancing and Model Reduction, (x) Nearness to Uncontrollability and Instability, (xi) Sensitivity and Conditioning for Eigenvalue Assignment; Lyapunov, Sylvester, and Riccati equations, (xii) H_2 and H_∞ Control, and (xiii) Selected Control Software.

In what follows, we give an overview of each of these topics with references to the Chapter(s) and Sections(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, along with the **conditioning** of these problems and **numerically stability** of the algorithms have been covered.

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.

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.

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. 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:

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

$$y(t) = Cx(t) + Du(t). \tag{1.2.2}$$

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 matrix exponential $e^{At} = I + At + (A^2t^2/2) + \cdots$ 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.

An obvious way to compute e^A is to use some simple canonical forms of A such as the *JCF* or a *companion form of A. It is shown in Chapter 5 by simple examples how such computations can lead to inaccurate results.* Computations using truncated Taylor series might also give erroneous result (see **Example 5.3.3**).

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**).

A method (**Algorithm 5.3.3**) due to Van Loan (1978) for computing an integral involving an matrix exponentials is also described in Section **5.3.5**.

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 short 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 Criterion (ii). Unfortunately, *this criterion does not yield to a numerically viable test for controllability (see* **Example 6.6.1***).*

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 6.7* (**Staircase Algorithm**).

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

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

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

Similarly, the discrete system

$$x(k+1) = Ax(k) \tag{1.4.2}$$

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 (i) 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 (ii) solving and testing the positive definiteness of the solution matrix X of the associated Lyapunov equations:

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

or

$$X - A^{\mathrm{T}}XA = M \text{ (for discrete-time stability).} \tag{1.4.4}$$

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 counterproductive in the sense that the most numerically viable method for solving a Lyapunov equation, namely, the **Schur** 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 dense system is to compute the eigenvalues of A using the universally used method, called the QR iteration with double shift* (**see Chapter 4 (Section 4.3.3)**).

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,* *is stated in Algorithm 7.5.1.* This method was later modified by Datta and Datta (1981). **According to theoretical operations-count, both these methods are about 3–4 times faster than the eigenvalue method and several times faster than the Lyapunov equation method.**

Two important inertia theorems (**Theorem 7.4.1** and **Theorem 7.4.2**) are stated in **Section 7.4**.

1.5 LYAPUNOV, SYLVESTER, AND ALGEBRAIC RICCATI EQUATIONS (CHAPTERS 8 AND 13)

The Lyapunov equations (1.4.3) and (1.4.4) arise in (i) Stability and Robust Stability Analyses (**Chapter 7**), (ii) Model Reduction (**Chapter 14**), (iii) Internal Balancing (**Chapter 14**), and (iv) Determining H_2-norm (**Chapter 7**).

A variation of the Sylvester equation $XA + BX = M$ called the **Sylvester-observer equation**, 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 JCF 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 20 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 JCFs of A. With A and C as chosen in **Example 8.5.1**, the *solution of* (1.4.3) *via diagonalization of A (available in MATLAB function* **lyap2***) is very different from the exact solution.*

The methods of choice are: (i) *The Schur method* (**Section 8.5.2**) *for the Lyapunov equation* and (ii) *The Hessenberg–Schur method* (**Algorithm 8.5.1**) *for the Sylvester equation.*

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 needed for **balanced realization** in the context of **model reduction (Chapter 14)**.

It is numerically desirable that L is found without explicitly computing the matrix X and without forming the matrix $C^{\mathrm{T}}C$ or BB^{T}. A method for obtaining such an L due to Hammarling 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):

$$XA + A^{\mathrm{T}}X - XBR^{-1}B^{\mathrm{T}}X + Q = 0 \tag{1.5.1}$$

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

$$A^{\mathrm{T}}XA - X - A^{\mathrm{T}}XB(R + B^{\mathrm{T}}XB)^{-1}B^{\mathrm{T}}XA + Q = 0 \tag{1.5.2}$$

and their variations arise in (i) LQR and LQG Design (**Chapters 10 and 12**), (ii) Kalman Filtering (**Chapter 12**), and (iii) 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

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

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). The Schur method for the CARE is described in **Algorithm 13.5.1.** Section 13.5.1 also contains some discussions on the Schur method for the DARE. The Schur method for the DARE is based on finding a stable invariant subspace of the associated symplectic matrix

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

The Schur methods, however, may not give an accurate solution in case R is nearly singular. This difficulty can be overcome by using an extended matrix pencil. The stabilizing solution of the CARE maybe computed by finding the *ordered generalized Schur decomposition* of this pencil using the QZ iteration algorithm. Such a method is called an *inverse-free generalized Schur method* and is described in **Algorithm 13.5.3** and **Algorithm 13.5.4**, respectively, for the CARE and DARE.

The **matrix sign-function methods** has been developed in **Section 13.5.3.** 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 (**Algorithm 13.5.7**), the matrix $H' = (P+N)^{-1}(P-N)$, where $P = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}$, and $N = \begin{pmatrix} I & S \\ 0 & A^{\mathrm{T}} \end{pmatrix}$ is computed first and then the matrix sign-function method for the CARE is applied.

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 **Algorithms 13.5.8** and **13.5.10**, while **Algorithms 13.5.9** and **13.5.11**, described **Newton's methods with line search** for the CARE and the DARE, respectively.

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 a 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 an MR if and only if it is both controllable and observable* (**Theorem 9.2.1**).

The two MRs are related via a nonsingular transforming matrix (**Theorem 9.2.2**) and the degree of an MR is called the **McMillan degree.**

The existing algorithms for finding an MR 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 ith 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.

Two subspace identification algorithms (**Algorithms 9.4.1 and 9.4.2**) *from DeMoor et al.* (1999), *that do not require explicit computations of Markov parameters, is presented in Section* 9.4.

Also stated in this chapter is a subspace algorithm (**Algorithm 9.4.3**) 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 methods are numerically stable practical methods for systems identification.

1.7 FEEDBACK STABILIZATION AND EIGENVALUE ASSIGNMENT (CHAPTERS 10 AND 11)

Suppose that the uncontrolled system (1.4.1) 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.

Computing the feedback vector f via controller canonical form or using the well-known Ackermann formula for single-input problem does not yield to a numerically viable algorithm (see **Example 11.2.1**).

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 (i) Recursive algorithms (**Algorithms 11.2.1 and 11.3.1**), based on evaluations of some simple recursive relations, (ii) QR and RQ type algorithms (**Algorithms 11.2.2, 11.2.3** and the one described in **Section 11.3.2**), (iii) The Schur algorithm (**Algorithm 11.3.3**) based on reduction of A to a real Schur form, and (iv) The robust EVA algorithm (**Algorithm 11.6.1**).

A parametric algorithm (**Algorithm 11.3.4**) for ***Partial eigenvalue assignment (PEVA)*** is described in **Section 11.3.4**. Lyapunov and Sylvester equations can also be used for feedback stabilization and EVA. 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 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 the single-input problem and* **Algorithm 11.3.3** *for the multi-input problem. Also, Algorithm 11.3.1 is extremely easy to use.*

1.8 STATE ESTIMATION (CHAPTER 12)

In many practical situations, the states are not fully accessible, but the designer knows the input $u(t)$ and the output $y(t)$. However, for stabilization and EVA 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

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

where the matrix K is constructed such that $A - KC$ is a stable matrix, then the error vector $e(t)$ has the property that $e(t) \to 0$ as $t \to \infty$. 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 EVA problem for the pair $(A^{\mathrm{T}}, C^{\mathrm{T}})$.

An alternative approach for state estimation is via solution of the Sylvester equation $XA - FX = GC$ (see **Theorem 12.3.1**).

Two numerically reliable algorithms (**Algorithms 12.7.1** and **12.7.2**) for the Sylvester-observer equation, 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 nonsingularity of the solution X of the Sylvester-observer equation have been given in **Theorems 12.6.1** and **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.

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 the 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 $T^{-1}C_G T^{-T} = T^T O_G T = \Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_d, \sigma_{d+1}, \cdots, \sigma_n)$, where C_G and O_G are, respectively, controllability and observability Grammians. The diagonal entries $\sigma_1, \cdots, \sigma_n$ 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. Let $G(s)$ and $G_R(s)$ denote the transfer function matrices, respectively, of the original and the reduced-order models. Then a bound for the error $E = \|G(s) - G_R(s)\|_\infty$ is given in **Theorem 14.4.1**.

An algorithm (**Algorithm 14.2.1**) for constructing a balanced realization, based on the SVD of the matrix $L_o^T L_C$, where L_o and L_C are, respectively, the Cholesky factors of observability and controllability Grammians, is given in Section 14.2. *The difficulty with this method is that the transforming matrix T may be ill-conditioned*(see **Section 14.2.2**). An algorithm, based on the Schur decomposition of the matrix $C_G O_G$, that overcomes this difficulty is the Schur algorithm, **Algorithm 14.4.2**. The Schur algorithm was developed by Safonov 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 NEARNESS TO UNCONTROLLABILITY AND INSTABILITY (CHAPTERS 6 AND 7) AND ROBUST STABILITY AND STABILITY RADIUS (CHAPTERS 7 AND 10)

1.10.1 Nearness to Uncontrollability and Instability

There are systems which are theoretically perfectly controllable, but may be very close to uncontrollable systems (see the Example in **Section 6.9**).

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 and Laub 1998) that

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

where σ_n denotes the smallest singular value. Several algorithms (Miminis 1981; Wicks and DeCarlo 1991; Elsner and He 1991) 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 10s 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**.

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

1.10.2 Robust Stability and Stability Radius (Chapters 7 and 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. Two robust stability results **(Theorems 7.7.1 and 7.7.2)** using Lyapunov equations are given in Chapter 7.

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

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

where $\bar{\sigma}(M)$, following the notation of Qiu *et al.* (1995), **denotes the largest singular value of** M (i.e., $\bar{\sigma}(M) = \sigma_{\max}(M)$). 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 radii are given, respectively, in **Theorems 7.8.1** and **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.

1.11 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 EVA problems (Sections 11.4 and 11.5).**

1.12 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 are stated in **Chapter 10** of this book. A simplified version of the output feedback H_∞-control problem and a result on the existence of a solution have been stated in **Section 10.6.3** of the chapter.

Solution of H_∞ Control Problems Requires Computation of H_∞-Norm.

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*

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

where $R = \gamma^2 I - D^{\mathrm{T}}D$, 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**:

$$\gamma_{lb} = \max\{\sigma_{\max}(D), \sigma H_1)\}$$

$$\gamma_{ub} = \sigma_{\max}(D) + 2\sum_{i=1}^{n} \sigma H_i,$$

where σ_{H_i} is the ith **Hankel singular value.** The Hankel singular value are the square-roots of the eigenvalues of the matrix $C_{\mathrm{G}}O_{\mathrm{G}}$, where C_{G} and O_{G} are, respectively, the controllability and observability Grammian.

1.13 SOFTWARE FOR CONTROL PROBLEMS

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

- 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 classrooms, because, students, researchers, and teachers will have an 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 Arnold and Laub (1984). for Riccati equations, HTOOLS for H_∞ and H_2 synthesis problems developed by Varga and Ionescu (1999), etc.

A brief description of some of these tool boxes appear in Appendix A. Some of the software packages developed by individuals may be obtained from the authors themselves. An internet search might also be helpful in locating these softwares.

References

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