The Truth about Conjugate Gradients

Victor Eijkhout
Texas Advanced Computing Center
The University of Texas at Austin
Austin, TX 78758
eijkhout@tacc.utexas.edu

June 3, 2016

Abstract

This paper brings together a bundle of facts about polynomial iterative methods for solving linear systems of equations.

Contents

1 **Executive summary** *6*
1.1 *Basics* *6*
1.2 *Computation of coefficients* *7*
2 **Introduction** *8*
3 **Definition and elementary properties of polynomial iterative methods** *8*
3.1 *Vector sequence notation and conventions* *8*
3.2 *Polynomial vector sequences* *10*
3.2.1 Basic statement *10*
3.2.2 Preconditioned polynomial methods *11*
3.2.3 Extended iterative methods *12*
3.3 *Krylov sequences* *13*
3.4 *Matrix formulation* *14*
3.5 *Residual sequences* *17*
3.5.1 Affine combinations of residual sequences *20*
3.6 *Termination of the iterative process* *22*
4 **Hessenberg matrices** *23*
4.1 *Hessenberg matrices with zero column sums* *23*
4.2 *QR decompositions of Hessenberg matrices* *25*
4.3 *Hessenberg matrices of residual sequences* *26*
5 **Characterisation of polynomial iterative methods** *28*
6 **Minimization** *29*
6.1 *Minimization of residuals in L_2 norm* *29*
6.2 *Minimization of residuals in the A^{-1} norm* *32*
6.3 *Minimization under general inner product* *33*
6.4 *Lanczos' Minimized iterations* *34*
6.5 *Line searches* *35*
7 **Orthogonalization** *36*
7.1 *Types of orthogonality* *36*
7.2 *Theoretical conditions on orthogonality* *37*

7.3 *Breakdown conditions* 39
7.3.1 Arnoldi orthogonalization under M^{-1} inner product; symmetric case 39
7.3.2 Arnoldi orthogonalization under M^{-1} inner product; nonsymmetric case 39
7.3.3 Arnoldi orthogonalization under AM^{-1} inner product 40
7.3.4 Lanczos orthogonalization 40
7.4 *General Arnoldi orthogonalization* 40
7.5 *Arnoldi orthogonalisation and the Conjugate Gradients method* 41
7.6 *Lanczos orthogonalisation and the BiConjugate Gradients method* 42
7.7 *Projection* 43
8 **The iterative formulation of the conjugate gradient and Lanczos methods** 44
8.1 *Search directions and the solution of linear systems* 44
8.2 *Derivation of scalars under Arnoldi and Lanczos orthogonalisation* 45
8.2.1 Coefficients under Arnoldi orthogonalization 46
8.2.2 Arnoldi orthogonalisation in the nonsymmetric case 47
8.2.3 Lanczos case 48
8.2.4 Lanczos bi-orthogonalisation 49
8.2.5 Algorithm for the BiConjugate Gradients method 50
8.2.6 Other definitions of search directions 50
8.2.7 Arnoldi case; AM^{-1} inner product 51
8.2.8 Arnoldi case; General orthogonalisation 51
8.2.9 Iterating under general inner product 52
8.3 *Three-term recurrences* 53
8.4 *Block Conjugate Gradients* 54
8.4.1 Preliminaries 54
8.4.2 Optimised algorithm 55
8.4.3 Block BiConjugate Gradients 56
8.5 *SYMMLQ* 57
9 **Minimum and quasi-minimum residual methods: MINRES, GMRES, QMR, TFQMR** 57
9.1 *Search directions in norm-minimising methods* 57
9.2 *Affine combinations of residuals* 58
9.3 *General theory of L_2 minimization methods* 60
9.3.1 Derivation of the coefficient minimization problem 60
9.3.2 Solution of the coefficient minimization problem 61
9.3.3 Solution vector update for minimization problems 61
9.3.4 Relation of residual norm to original iterations 62
9.3.5 Stagnation of the minimised iterations 62
9.4 *MINRES: The Minimum Residual method* 62
9.5 *GMRES: The Generalised Minimum Residual method* 63
9.5.1 The Saad and Schultz algorithm 63
9.5.2 Breakdown of GMRES 63
9.5.3 Implementation of the QR decomposition 63
9.5.4 Direct computation of the GMRES vectors 63
9.5.5 Left-preconditioned GMRES 64
9.6 *Residual-minimising Arnoldi methods for the non-symmetric case: GCR, Orthodir, GMRESr* 64
9.7 *Quasi-minimisation* 65
9.8 *QMR* 66
9.9 *Residual smoothing* 66
9.10 *TFQMR* 70
10 **Polynomial squaring methods** 70
10.1 *Matrix derivation of polynomial squaring methods* 70

10.2 *Coefficients in general Polynomial squaring methods* 71
10.3 *Standard presentation of Polynomial Squaring methods* 73
10.3.1 Vector recurrences 74
10.3.2 Work estimates 75
10.3.3 Computation of coefficients in CGS 75
10.3.4 Full CGS algorithm 75
10.3.5 Computation of coefficients in BiCGstab 76
10.4 *Three-term recurrence derivation Conjugate Gradient Squared* 76
10.4.1 Computation of coefficients in CGS 77
10.4.2 Computation of coefficients in BiCGstab 77
11 **Arnoldi methods** 78
11.1 *The Arnoldi algorithm* 78
11.2 *Characterisation of the Arnoldi sequence* 79
11.3 *Orthogonalisation of the Krylov sequence* 80
11.3.1 Gram-Schmidt and modified Gram-Schmidt in Arnoldi methods 80
11.3.2 After-the-fact orthogonalisation 81
11.4 *Reduction to Hessenberg form by Householder reflections* 82
11.4.1 Basic ideas of Householder reflectors 82
11.4.2 Relation to Lanczos vectors 83
11.4.3 Retrieving Householder vectors from the Lanczos basis 84
11.5 *Implicitly restarted Arnoldi* 85
12 **Non orthogonality-based methods** 85
12.1 *Iterative refinement* 85
12.1.1 Propagation of error and residual 85
12.2 *Rewrites of Stationary Iteration* 86
12.3 *Stationary Iteration with varying solver* 87
12.4 *Convergence of stationary iterative methods* 88
12.5 *Parametrisation: Richardson iteration* 89
12.6 *Steepest descent* 89
12.7 *Recovering CG from Stationary Iteration* 90
12.8 *Chebyshev semi-iteration* 91
13 **Polynomials** 91
13.1 *Recap of earlier results* 91
13.2 *Orthogonal polynomials* 92
13.3 *CG polynomials* 93
13.4 *Polynomials of search directions* 94
13.5 *Constructing a sequence from a polynomial* 95
13.6 *Chebyshev polynomials* 95
14 **Convergence theory** 97
14.1 *Speed of convergence of Conjugate Gradients* 97
14.2 *Error reduction with isolated eigenvalues* 98
14.3 *Stopping tests* 98
14.3.1 Backward error analysis 98
14.3.2 Eigenvalue based estimates 99
14.4 *Breakdown conditions* 100
14.4.1 General theory 100
14.4.2 Iterative forms of the breakdown condition 101
15 **Eigenvalues** 101
15.1 *Ritz values and Harmonic Ritz values* 101
15.2 *Eigenvalues of the Hessenberg matrix* 102
15.2.1 Relationship between eigenvalues of A and H 102
16 **Other aspects of the iterative solution process** 102
16.1 *Inventory of all upper triangular matrices* 102
16.2 *The symmetrisable case* 103
16.3 *Iterative solution as solving reduced systems* 104
16.4 $AV_n = V_nH + something$ 105
16.5 *Singular matrices* 106
16.6 *Iterating in subspaces* 106

16.6.1 approach 1 *106*
16.6.2 approach 2 *107*
16.6.3 approach 3 *107*
16.6.4 Null spaces and preconditioners *107*
16.6.5 approach 4 *107*
16.7 *Shift invariance* *108*
16.8 *Equivalent formulas for the scalar quantities* *109*
16.8.1 Elimination of the $r_i^t r_i$ inner product *109*
16.8.2 Elimination of the $p_i^t A p_i$ inner product *110*
16.8.3 Other approaches for minimizing synchronisation overhead *111*
16.8.4 Pipelined CG *112*
16.9 *Complex symmetric CG* *113*
16.10 *Double size systems* *114*
16.10.1 Singular value computations *114*
16.10.2 Iterating on the normal equations; LSQR *114*
16.10.3 Lanczos iteration as augmented Conjugate Gradient iteration *114*
16.11 *Scaled sequences* *116*
16.11.1 Theory *116*
16.11.2 A normalised CG algorithm *117*
16.11.3 Normalised CGS *119*
16.12 *Seed systems* *121*
16.13 *Polynomial acceleration* *121*
16.14 *Left and right preconditioning* *122*
16.14.1 Left preconditioning *122*
16.14.2 Right preconditioning *124*
16.14.3 Two-sided preconditioning *125*
16.14.4 The Eisenstat trick *126*
16.15 *Preconditioning by the symmetric part and separable approximations* *127*
16.16 *Diagonal coefficient matrices* *127*
16.17 *CG in Function Spaces* *128*
17 **History** *131*
A **Matlab code examples** *136*
A.1 *Minimum residual methods* *136*
A.1.1 GMRES *136*
A.2 *Polynomial squaring methods* *137*
A.2.1 Conjugate Gradient Squared *137*
A.3 *Arnoldi method* *137*
A.4 *Householder reflectors* *138*
A.4.1 Basic reduction routines *138*
A.4.2 Retrieving Householder vectors from a Lanczos basis *138*
A.5 *Utilities* *138*
A.5.1 Orthogonalisation *138*

Lemma 4: Preconditioned methods are polynomial methods *11*
Lemma 5: Updating a sequence equivalent to making from scratch *15*
Lemma 7: Polynomial application vs combinations of a Krylov sequence *15*
Lemma 8: Update / generation relations for polynomial sequence *16*
Lemma 9: Residual sequences consist of normalised combinations of Krylov sequence *17*
Corollary 12: Iterative method can be updated with residuals *18*
Lemma 13: Residuals satisfy update relation *18*
Lemma 17: Upper triangular matrix for X is subblock of upper triangular matrix for R *19*
Lemma 18: Affine combinations of a residual sequence are a residual sequence *20*
Lemma 19: All residual sequences from r_1 are affine combinations of each other *21*
Lemma 20: Residual sequences can be updated with each other *21*
Lemma 23: H zero column sums iff U normalized *23*
Lemma 25: $H = (I-J)U$ iff zero column sums *24*
Lemma 26: $U(I-J) = (I-J)V$ if U constant column sums, **this only works for H square** *24*
Lemma 28: QR decomps of equiv Hess related by two bidiagonal mats *25*
Lemma 29: In $AM^{-1}R = RH$, H Hessenberg iff R is combination of Krylov sequence *26*
Theorem 30: In $AM^{-1}R = RH$, H zero column sum Hessenberg iff R residual sequence *27*
Corollary 33: Left-preconditioned residual equation *29*
Corollary 36: Minimisation of residuals $\Leftrightarrow$ AM^{-1}-orthogonal residuals *30*
Theorem 37: AM^{-1}-orthogonal R is minimal *31*
Corollary 41: For A spd, M^{-1}-orthogonal R is minimal in A^{-1} norm *32*
Algorithm 1: Semi-orthogonalise Krylov sequence to arbitrary sequence *37*
Lemma 46: Existence of general Arnoldi orthogonalization *40*
Algorithm 2: Bi-orthogonalisation *42*
Theorem 51: BiCG orthogonalisation *42*
Algorithm 3: Conjugate Gradients Method *47*
Algorithm 4: BiConjugate Gradients Method *50*
Algorithm 5: Block Conjugate Gradient Method, version 1 *55*
Algorithm 6: Block Conjugate Gradient Method, version 2 *55*
Corollary 56: Hessenberg matrices from methods for the same problem are similar *59*
Algorithm 7: Conjugate Gradients Squared *76*
Algorithm 8: Arnoldi reduction to Hessenberg form *78*
Lemma 63: Uniqueness of Arnoldi method *79*
Derive SSOR from forward/backward SOR *87*
Lemma 65: Convex combinations of iterates same as residuals *90*
Lemma 66: Orthogonal polynomials satisfy three-term recurrence *92*
Lemma 67: Residual sequences generated by symmetric matrix satisfy three-term recurrence *93*
Lemma 68: Residual polynomials satisfy recurrence with coefficients in H *94*
Lemma 69: Polynomials for residuals and search directions *94*
Lemma 71: Convergence is dominated by estimate of polynomial on spectrum *97*
Theorem 72: Convergence of CG determined by best polynomial bound on spectrum *97*
Theorem 73: Backward error can be computed from residual and solution *99*
Lemma 74: Arnoldi orthogonal R constructable from independent K *100*
Lemma 75: Lanczos orthogonal R exists if K independent and no accidental $s_i^t r_i = 0$ *100*
Algorithm 9: Symmetrised Conjugate Gradient Method *103*
Algorithm 10: Scale combination of Krylov sequence to residual sequence *116*
Lemma 77: Condition for scaling a sequence to a residual sequence *116*

Theorem 78: An orthogonal sequence can be scaled to a residual sequence *117*
Algorithm 11: Conjugate Gradient algorithm with normalised matrix-vector product *119*
Algorithm 12: Left preconditioned BiConjugate Gradient Method *123*
Algorithm 13: Right Preconditioned BiConjugate Gradient Method *124*

1 Executive summary

Previous section: **??** Next section: 2

1.1 Basics

Let $A\bar{x} = b$. The Cayley-Hamilton theorem states that there is a polynomial such that $A^{-1} = -\pi(A)$. Let x_1 be an approximation to $\bar{x}$ and $r_1 = Ax_1 - b$, then $\bar{x} - x_1 = \pi(A)r_1$. Since $\bar{x}$ is also the solution of any system $M^{-1}A\bar{x} = M^{-1}b$, we generalise this as

$$\forall_M : \bar{x} - x_1 = \pi(M^{-1}A)M^{-1}r_1. \tag{1}$$

which leads to the definition of a *preconditioned polynomial iterative method*:

$$x_{i+1} - x_i = \pi_i(M^{-1}A)M^{-1}r_1. \tag{2}$$

The basic theorem of polynomial iterative methods states that R is the sequence of residuals of a preconditioned polynomial iterative method iff

$$AM^{-1}R = RH \tag{3}$$

where H is a Hessenberg matrix with zero column sums.

The class of conjugacy-based methods satisfies equation (**??**) and equation (**??**); coefficients follow from an orthogonality requirement on R. For *Arnoldi method* this is $(R,R)_B = 0$ for some B; for *Lanczos methods* it is $(R,S)_B = 0$ where S is another sequence.

1.2 Computation of coefficients

Since H with zero column sums can be factored $H = (I-J)D^{-1}(I-U)$ (with J the unit lower subdiagonal), we split equation (3) as

$$APD = R(I-J), \qquad M^{-1}R = P(I-U) \tag{4}$$

by introducing search directions P.

Arnoldi The coefficients of D and U follow from the requirement that $R^tM^{-1}R$ be diagonal ($\Omega \equiv R^tM^{-1}R$). From

$$R^tM^{-1}R = R^tP(I-U) \tag{5}$$

we conclude that R^tP is upper triangular. From

$$P^tAPD = P^tR(I-J) \tag{6}$$

we then get that P^tAP is lower triangular. The coefficients of U then follow from

$$I-U = (R^tP)^{-1}\Omega \tag{7}$$

Define $\Theta = \mathrm{diag}(P^tAP)$ and take the diagonal of equation (6) to get $D = \Theta^{-1}\Omega^t$.

CG In the symmetric case we have $\Theta = P^tAP$, so from equation (6) we get $\Omega^t = \Theta D = P^tR(I-J)$. Use this in equation (7) to get

$$I-U = \Omega^{-1}(I-J^t)\Omega. \tag{8}$$

BiCG Suppose we have a Krylov method based on A^t, written as

$$A^tM^{-t}S = SH \Rightarrow A^tQD = S(I-J),\ M^{-t}S = Q(I-U) \tag{9}$$

The orthogonality requirement that $\Omega = S^tM^{-1}R$ be diagonal gives that S^tP and R^tQ are upper triangular, and from the $M^{-1}R = \ldots$, $M^{-t}S = \ldots$ equations $\mathrm{diag}(S^tP) = \mathrm{diag}(Q^tR) = \Omega$.

Furthermore, by

$$Q^tAPD = Q^tR(I-J), \qquad P^tA^tQD = P^tS(I-J)$$

we find that $\Theta = Q^tAp$ is both lower and upper triangular, hence diagonal. This gives again $\Theta D = \Omega$.
Combining

$$\Omega = \Theta D = Q^tR(I-J) \quad \text{and} \quad \Omega^t = R^tQ(I-U)$$

gives again

$$I-U = \Omega^{-t}(I-J^t)\Omega^t. \tag{10}$$

2 Introduction

Previous section: A Next section: 3

Many articles have already been written about the conjugate gradient method and the Lanczos algorithm. Some of these focus on preconditioners, some treat different variants of the basic method, and others derive error or convergence speed properties of the methods In this article we will give a unified presentation of the conjugate gradient method and the Lanczos algorithm, and derive properties regarding orthogonality and equivalence of various formulations. We will stress showing what the minimal assumptions are for various properties.

In general we will only be concerned with qualitative properties of the methods, leaving all quantitative results, such as convergence speed, aside. The conjugate gradient and Lanczos methods will here be derived as orthogonalization methods for Krylov sequences, and minimization properties will be derived from this orthogonality. Also, the fact that such methods can be used for the iterative solution of linear systems will be presented as a corollary rather than as a starting point of the discussion.

Additionally, the presentation here will differ from most others in the literature (with the exception of [42]) in that it is given in terms of matrices instead of vector sequences. Although an occasional part of the discussion may feel somewhat forced by this, in general the presentation is very concise.

3 Definition and elementary properties of polynomial iterative methods

Previous section: 2 Next section: 4

In this section we will define polynomial iterative methods, that is, sequences of vector iterates. Ultimately we will use such sequences to solve linear systems with.

3.1 Vector sequence notation and conventions

We are investigating vector sequences, that is, sequences

$$X \in \mathbf{R}^{n\times k}\colon X \equiv [x_i]_{i,i<k} \equiv i \mapsto x_i \qquad x_i \in \mathbf{R}^n,\ i = 0\ldots k,$$

where we use $[\langle\text{expression with } i\rangle]_i$ to denote a sequence indexed by i.

There are two types of operations on sequences. If we apply an operator

$$A \in \mathbf{R}^{n\times n}$$

from the left to a sequence X, then AX is the sequence $[Ax_i]_i$. We can also apply operators

$$U \in [\mathbf{R}^{n\times k} \to \mathbf{R}^{n\times k}]$$

from the right: if X is a sequence, XU is one too.

Since our vector sequences model iterative processes, and we can not look into the future, operators applied from the right will always be upper triangular or banded lower triangular.

We introduce two sets of upper triangular transformations:

Definition: Upper triangular and normalized upper triangular transformations

Definition 1

$$U \in \mathbf{U} \equiv i > j \rightarrow u_{ij} = 0;$$

$$U \in \mathbf{U^{(1)}} \equiv U \in \mathbf{U} \textit{ and } u_{1j} \equiv 1.$$

We also introduce sets of polynomials that will be seen to be closely related to these upper triangular transformations:

Definition: Polynomials and normalized polynomials

Definition 2

$$\mathbf{P^{(n)}} = \{\textit{polynomials of degree } n\};$$

$$\mathbf{P^{(n,1)}} = \{\textit{polynomials } P \textit{ of degree } n \textit{ such that } P(0) = 1\}.$$

When only an initial part of a sequence is needed we denote it by, for instance, X_n for the first n columns of X. An initial part of a right-multiplying operator is denoted by, for instance, U_n if square, or $H_{[n+1,n]}$ if rectangular. In general, however, the simplicity of presentation will benefit greatly from the fact that we formulate most everything in terms of infinite sequences. Section 16.4 will address some of the subtleties involved.

Block statement
All of the above can be generalised to the block case, where $x_i \in \mathbf{R}^{n\times k}$, represented as a block of k vectors. The upper triangular matrices then become of block upper triangular shape, where the blocks corresponding to the scalar elements are of size $k \times k$. Often, the element blocks are equal to a scalar multiple of the identity matrix; we can then use the outer product notation $A \otimes I_k$.

Block iterative methods are treated extensively in section 8.4; however we will remark in relevant cases on block extensions of statements we prove. In preparation for these extensions, even scalar statements will often be written in such a way that commutativity of scalars is not used.
End of block statement

3.2 Polynomial vector sequences

In this section we introduce the basic form of iterative methods. All methods in this monograph will conform to this scheme.

3.2.1 Basic statement

Definition: Polynomial iterative method

Definition 3 *A polynomial iterative method is a sequence of vectors* $\{x_i\}_{i\geq 1}$ *in* $\mathbf{R}^N$[1], *denoted by a 4-tuple*

$$X = P\langle\{\pi_i\}_{i\geq 1}, A, x_0, f\rangle,$$

where x_0, f *are vectors in* $\mathbf{R}^N$, A *is an* $N \times N$ *matrix, and* $\pi_n \in \mathbf{P}^{(\mathbf{n})}$; *the sequence* X *is defined by*

$$x_{i+1} - x_0 = \pi_i(A)\{Ax_0 - f\}. \tag{11}$$

Block statement
In a block version of this definition, x_1 and f are elements of $\mathbf{R}^{n\times k}$, and the polynomial has matrix-valued coefficients in $\mathbf{R}^{k\times k}$.
End of block statement

Definition (11) can be motivated informally from the following observations. Let $Ax = f$ and let x_0 be an arbitrary vector. First of all

$$r_0 = Ax_0 - f \Rightarrow x = A^{-1}f = x_0 - A^{-1}r_0,$$

stating that we can find the solution x if we can compute $A^{-1}r_0$.

Then, by the Cayley-Hamilton theorem, there is a polynomial ϕ such that $\phi(A) = 0$, and without loss of generality we can write $\phi(x) = 1 + x\pi(x)$[2] with π another polynomial. Then

$$A^{-1} = -\pi(A) \qquad \text{so} \qquad x - x_0 = \pi(A)r_0.$$

Polynomial iterative methods then can be viewed as constructing successive polynomials that in some sense approximate this polynomial π.

Remark 1 *In practice, we often encounter the formulation*

$$x_{i+1} - x_i = \pi_i(A)\{Ax_0 - f\}.$$

It is not hard to see that this is equivalent to equation (11). We will go into this more in section 3.4, in particular lemma 5.

We sharpen the above statement slightly, since we do not actually need $\phi(A) = 0$: it suffices if ϕ is such that $\phi(A)r_0 = 0$[3].

Theorem 2 *Let polynomials* π *and* ϕ *be related as* $\phi(x) = 1 + x\pi(x)$, *then*

$$\phi(A)r_0 = 0 \text{ iff } x_0 + \pi(A)r_0 = A^{-1}f.$$

The Cayley-Hamilton theorem gives an upper bound on the degree of the polynomial needed to compute A^{-1}. In some cases, a polymial of lesser degree may already yield the inverse in exact arithmetic, or a sequence of polynomials may yield progressively better approximations to the system solution, so that early termination of the process is possible. For more about termination, see section 3.6; for convergence, see section 14 .

1. Since we will mostly deal with vectors of the same size, the vector dimension N will go unremarked on.
2. We will see this equation reflected in equation (24).
3. That said, I am not aware of any methods that exploit the difference.

3.2.2 Preconditioned polynomial methods

In the definition of a polynomial vector sequence we call $x = A^{-1}f$ the *solution vector* of the method. Now, there are many choices for the matrix A and vector f that all have x as solution. This raises the question what the relation is between a given method and these other systems.

Lemma 3 *Let X be the polynomial method $P\langle\{\pi_i\}_{i\geq 1}, \bar{A}, x_0, \bar{f}\rangle$ where $\bar{A}x = \bar{f}$ for the solution x, and let a matrix A and a vector f be such that also $Ax = f$, then there is a matrix M such that*

$$x_{i+1} - x_0 = \pi_i(M^{-1}A)M^{-1}(Ax_0 - f). \tag{12}$$

Proof: Let X be the method $P\langle\{\pi_i\}_{i\geq 1}, \bar{A}, x_0, \bar{f}\rangle$ where $\bar{A}x = \bar{f}$. Let A and f be such that also $Ax = f$. Then

$$A^{-1}f = \bar{A}^{-1}\bar{f} \Rightarrow \bar{f} = \bar{A}A^{-1}f.$$

Now define $M^{-1} = \bar{A}A^{-1}$, then

$$\bar{A} = M^{-1}A, \quad \bar{f} = M^{-1}f,$$

so

$$x_{i+1} - x_0 = \pi_i(M^{-1}A)M^{-1}r_0$$

where $r_0 = Ax_0 - f$. •

Equation (12) can also be written as

$$x_{i+1} - x_0 = M^{-1}\pi_i(AM^{-1})(Ax_0 - f).$$

Equation (12) is not the description of a polynomial sequence as defined above. However, it is common enough that we will give it a name.

Definition: Preconditioned polynomial method

Definition 4 *A sequence generated by equation (12) is called a (left) preconditioned polynomial method for the solution $A^{-1}f$; we use the notation $P\langle\{\pi_i\}_{i\geq 1}, A, M, x_0, f\rangle$.*

The matrix M is called a 'preconditioner', since it will be intended as a conditioning operator for the original system. Note that M need not be – and in many practical applications is not – given explicitly: the only requirement is that the action $y \leftarrow M^{-1}x$ is computable, that is, it should be possible to solve y from $My = x$.

We need to tie preconditioned methods to the polynomial methods we already defined.

Preconditioned methods are polynomial methods

Lemma 4 *Let $Ax = f$ and nonsingular M be given and let*

$$\{x_i\}_i = P\langle\{\pi_i\}_{i\geq 1}, A, M, x_0, f\rangle$$

be a preconditioned polynomial method, then $\{x_i\}$ is a polynomial iterative method with solution x:

$$P\langle\{\pi_i\}_{i\geq 1}, A, M, x_0, f\rangle = P\langle\{\pi_i\}_{i\geq 1}, M^{-1}A, x_0, M^{-1}f\rangle \tag{13}$$

Proof: Define $\bar{A} = M^{-1}A$ and $\bar{f} = M^{-1}f$, then $\bar{A}x = \bar{f}$. Now write the definition of the preconditioned polynomial method as in equation (12):

$$x_{i+1} - x_0 = \pi_i(M^{-1}A)M^{-1}(Ax_0 - f)$$

then we rewrite this as

$$x_{i+1} - x_0 = \pi_i(\bar{A})(\bar{A}x_0 - \bar{f}),$$

which conforms to the definition of a polynomial iterative method, and equation (13) follows. •

Equation (12) is called a 'left preconditioned' iterative method, since it is a polynomial method based on the left-transformed system

$$M^{-1}Ax = M^{-1}f.$$

We can derive left preconditioned methods from the Cayley-Hamilton theorem by observing that

$$\begin{aligned} \bar{x} = A^{-1}f &= (M^{-1}A)^{-1}(M^{-1}f) = (M^{-1}A)^{-1}M^{-1}(Ax_0 - r_0) \\ &= x_0 - (M^{-1}A)^{-1}M^{-1}r_0 \\ &= x_0 + \pi(M^{-1}A)M^{-1}r_0 \end{aligned}$$

So far we talked about left preconditioning which corresponds to a scaling of the system. Right preconditioning, which is more like a scaling of the sequence of iterates, can be derived as follows:

$$Mx_{i+1} - Mx_0 = \pi_i(AM^{-1})(AM^{-1}(Mx_0) - f).$$

In other words, $MX = P\langle\{\pi_i\}_{i\geq 1}, AM^{-1}, Mx_0, f\rangle$, which means that MX is a polynomial method for the right preconditioned system

$$AM^{-1}x = f.$$

Writing $Y = MX$, we then have a polynomial method $Y = P\langle\{\pi_i\}_{i\geq 1}, AM^{-1}, Mx_0, f\rangle$ for the solution $MA^{-1}f = Mx$, and we can reconstruct a method for x by $x_i = M^{-1}y_i$.

Constructing $y_0 = Mx_0$ is not feasible in general, since we usually can only compute the action of M^{-1} on some vector. However, we can simply start iterating with an arbitrary y_0, and let x_0 be defined implicitly from this relation.

The final iterate x_n can still be found from $x_n = M^{-1}y_n$ where y_n is the final iterate of the right preconditioned method we are actually computing. There is a practical disadvantage to proceeding this way: if we monitor the iteration process, we can not relate the y_i vectors to the original system and solution without incurring extra work for the transformation $x_i = M^{-1}y_i$.

More about left and right preconditioning in section 16.14.

Extended statement

3.2.3 Extended iterative methods

Although the vast majority of iterative methods, both in this monograph and in practice, are of the polynomial form of equation (11) or equation (12), sometimes we come across more general methods, where the x_i vectors are updated with arbitrary – or not-so-arbitrary but at least not polynomially derived – vectors. For this we define

Definition: Extended (preconditioned) iterative methods

Definition 5 *A sequence $\{x_i\}$ is called an extended iterative method if*

$$x_{i+1} - x_i \in [\![w_0, \ldots, w_i]\!];$$

it is called an extended preconditioned iterative method if

$$x_{i+1} - x_i \in M^{-1}[\![w_0, \ldots, w_i]\!].$$

Since, in practice, iterative methods are almost always preconditioned, the first, unpreconditioned, form is not so much a more basic form, as it is a form where a, possibly iteration-dependent, preconditioner is incorporated into the update vector.

In the rest of this monograph we will occasionally remark on the extended form of certain statements.
End of extended statement

3.3 Krylov sequences

Polynomial methods are tightly connected, in ways that we will explore later, to so-called Krylov sequences.

Definition: Krylov sequence

Definition 6 *Let a matrix A, a sequence X and a vector f be given. We define the Krylov sequence with respect to A, x_0, and f as*

$$K\langle A, x_0, f\rangle \equiv k_0 = Ax_0 - f, \quad k_{i+1} = Ak_i,$$

or, shorter, with respect to just A and a starting vector k_0:

$$K\langle A, k_0\rangle \equiv k_{i+1} = Ak_i.$$

An initial part K_n of a Krylov sequence K may span an n-dimensional subspace, or it may be invariant. If $k_{n+1} \in [\![k_0, \ldots, k_n]\!]$, then K_n is invariant: $AK_n \subset K_n$. Invariant subspaces can be good or bad; theorem 21 shows that an invariant subspace signifies that the solution of a linear system is reached. Later we will see that in some methods invariant subspaces mean trouble.

Question: *Find the reference*

Computationally, generating the Krylov sequence is not a good way of finding the subspace spanned by K_n. The vectors k_i will tend more and more towards the dominant eigenvector of A. The iterative methods presented later in this monograph are essentially ways of generating the same basis as the Krylov sequence, but in a more stable manner.

3.4 Matrix formulation

In this section we will develop the matrix formulation tools that will facilitate further presentation and analysis of polynomial iterative methods.

First of all, we often abbreviate vector sequences as a matrix:

$$X = (x_0, x_2, \ldots).$$

Next we introduce the 'left-shift' operator J for sequences:

$$J = (\delta_{i,j+1}) = \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & 1 & \ddots & \\ & & \ddots & 0 \\ & & & 1 \end{pmatrix}$$

so that for sequences X and Y the statement $\hat{Y} = XJ$[4] implies $y_i = x_{i+1}$. Also, Krylov sequences $y_{i+1} = Ay_i$ can be denoted as $A\hat{Y} = YJ$. Furthermore, we introduce the matrix

$$E_1 = \begin{pmatrix} 1 & \ldots \\ 0 & \ldots \\ \vdots & \end{pmatrix}$$

which picks the first element of a sequence: $Y = XE_1$ is shorthand for '$y_i = x_0$ for all i'.

Block statement
In block versions of these definitions, the scalars 1 are replaced by identity matrices, and the zeros by zero matrices, all of size $k \times k$, where k is the width of a block vector, so $E_1 \rightarrow E_1 \otimes I_k$ and $J \rightarrow J \otimes I_k$.
End of block statement

The operators $J - I$ and $J - E_1$ are convenient in talking about updating a sequence. We will use the notational convention that $J - I$ and $J - E_1$ will always be rectangular matrices with one more row than columns; to prevent notational overload their exact sizes will not be indicated, unless they are not clear from the context.

We can now summarize sequence updates in block form:

$$Y_n = X_{n+1}(J - I) \qquad \Leftrightarrow \qquad y_i = x_{i+1} - x_i \tag{14}$$

and

$$Y_n = X_{n+1}(J - E_1) \qquad \Leftrightarrow \qquad y_i = x_{i+1} - x_0. \tag{15}$$

The relation between $J - I$ and $J - E_1$ is as follows:

$$\begin{array}{rcl} (J - E_1)_{[n+1,n]} & = & (J - I)_{[n+1,n]}(I - J^t)^{-1}_n \\ (J - I)_{[n+1,n]} & = & (J - E_1)_{[n+1,n]}(I - J^t)_n \end{array} \tag{16}$$

The following auxiliary lemma shows that constructing a sequence by updating it from its first element

$$x_{i+1} - x_0 = \sum_{j \leq i} k_j c_{ji}$$

is equivalent to updating it from the previous element as

$$x_{i+1} - x_i = \sum_{j \leq i} k_j \tilde{c}_{ji}.$$

Updating a sequence equivalent to making from scratch

4. Recall the notation that $\hat{Y}$ is the sequence Y minus its last column.

Lemma 5 *If X and K are sequences, U is upper triangular, then*

$$X_{n+1}(J-E_1) = KU_n \qquad \text{iff} \qquad X_{n+1}(J-I) = KV_n$$

for some upper triangular matrix V.

Proof: Let $X(J-E) = KU$, then using equation (16):

$$\begin{aligned} X(J-I) &= X(J-E)(I-J^t) \\ &= KU(I-J^t) = KV \end{aligned}$$

where $V = U(I-J^t)$. Conversely, if $X(J-I) = KV$:

$$\begin{aligned} X(J-E) &= X(J-I)(I-J^t)^{-1} \\ &= KV(I-J^t)^{-1} = KU. \end{aligned}$$

•

Block statement
There are two block versions of this lemma, replacing all ones by identity blocks: U can be either a true upper triangular matrix, or a block triangular matrix where the diagonal blocks are not necessarily themselves of diagonal form.
End of block statement

Remark 6 *In the proof of lemma 5 we remarked that the upper triangular matrices are related by*

$$V = U(I-J^t).$$

In other words,

$$v_{*,1} = u_{*,1}, \qquad j > 1 \rightarrow v_{*,j} = u_{*,j} - u_{*,j-1}$$

and conversely

$$j > 1 \rightarrow u_{*j} = v_{*1} + \cdots + v_{*,j}.$$

The right hand side in equation (12) can be described differently in terms of a Krylov sequence: applying successive polynomials to an initial vector is equivalent to taking linear combinations of the Krylov series.

Polynomial application vs combinations of a Krylov sequence

Lemma 7 *Let a matrix A, a preconditioner M, a vector k_0, and polynomials π_i with $\deg(\pi_i) = i-1$ be given, then*

$$[\pi_i(M^{-1}A)M^{-1}k_0]_i = K\langle M^{-1}A, M^{-1}k_0\rangle U(\pi),$$

where $U(\pi)$ is an upper triangular matrix with the columns containing the polynomial coefficients; specifically,

$$\pi_i(x) = u_{ii}x^{i-1} + \cdots + u_{2i}x + u_{1i}.$$

Block statement
For the block version of this lemma we have to note that the polynomial application proceeds as

$$A^i k_0 u_{ii} + \cdots + Ak_0 u_{2i} + k_0 u_{1i},$$

where the u_{ij} are $k \times k$ blocks. Correspondingly, U is a block upper triangular matrix.
End of block statement

We now have the following matrix characterisation of polynomial iterative methods:

Update / generation relations for polynomial sequence

Lemma 8 *Let X be a sequence, and let the matrix A and the vector f be given. Define $k_0 = Ax_0 - f$, and let $K = K\langle M^{-1}A, M^{-1}k_0\rangle$. Then the following statements are equivalent.*

1. $X = P\langle\{\pi_i\}_{i\geq 1}, A, M, x_0, f\rangle$
2. *There are polynomials $\{\pi_i\}_{i\geq 1}$ such that*

$$X_{n+1}(J-E_1) = (\pi_1(M^{-1}A)M^{-1}k_1, \pi_2(M^{-1}A)M^{-1}k_1, \ldots)_n.$$

3. *There are polynomials $\{\pi_i\}_{i\geq 1}$ such that*

$$X_{n+1}(J-I) = (\pi_1(M^{-1}A)M^{-1}k_1, \pi_2(M^{-1}A)M^{-1}k_1, \ldots)_n.$$

4. *There is an upper triangular matrix U such that*

$$X_{n+1}(J-E_1) = KU_n.$$

5. *There is an upper triangular matrix V such that*

$$X_{n+1}(J-I) = KV_n.$$

The polynomials in items 2 and 3 are related by

$$\phi_{*,1} = \pi_{*,1}, \quad j > 1 \rightarrow \begin{cases} \phi_{*,j} = \pi_{*,j} - \pi_{*,j-1} \\ \pi_{*j} = \phi_{*1} + \cdots + \phi_{*,j}. \end{cases} \tag{17}$$

The upper triangular matrices in items 4 and 5 are related by

$$v_{*,1} = u_{*,1}, \quad j > 1 \rightarrow \begin{cases} v_{*,j} = u_{*,j} - u_{*,j-1} \\ u_{*j} = v_{*1} + \cdots + v_{*,j}. \end{cases} \tag{18}$$

Proof: Statements 1 and 2 are equivalent by definition 3. Lemma 7 shows the equivalence of 3 and 5, and of 2 and 4. Lemma 5 shows the equivalence of 5 and 4. The relation equation (18) was stated in remark 6 and the relation equation (17) between polynomials follows from it. •

This lemma states that polynomial methods, which by definition work by updating with a polynomial time the original residual, can also be considered as updating with combinations of a Krylov sequence.

Extended statement
The extended form of the block formulation gives us

$$X(J-I) = WU, \qquad X(J-E_1) = WV,$$

and

$$X(J-I) = M^{-1}WU, \qquad X(J-E_1) = M^{-1}WV,$$

for preconditioned extended methods.
End of extended statement

3.5 Residual sequences

In the previous section we looked at vector sequences. The intended model for these is that of successive approximations to the solution of a linear system. Often, the residuals corresponding to these iterates are more interesting to consider.

With the vector $e = (1,\ldots)^t$ we can denote residuals $r_i = Ax_i - f$ as a sequence by $R = AX - fe^t$.

Definition: Residuals of a sequence

Definition 7 *Let a matrix A, a sequence X and a vector f be given. We define the residuals of the sequence X with respect to A and f as:*

$$R\langle A,X,f\rangle \equiv [r_i = Ax_i - f]_i.$$

Block statement
Block residuals are defined analogously, where f is a block vector of k columns, and e is a vector of $k \times k$ identity matrices.
End of block statement

It is an interesting fact that often we need not concern ourselves with the right hand side vector f, since it usually drops out of equations such as

$$\begin{aligned} R(J-I) &= AX(J-I) \\ R(J-E_1) &= AX(J-E_1) \end{aligned}$$

Definition: Residual sequence

Definition 8 *Let a matrix A a vector f and a sequence X be given, and let $R = R\langle A,X,f\rangle$ (definition 7). We call R a '(preconditioned) residual sequence' if X is a (preconditioned) polynomial iterative method.*

Lemma 8 above states that polynomial iterative methods use combinations of a Krylov sequence for updating. The following lemma shows that the residuals of the iterative method are then themselves combinations of this Krylov sequence; there is a normalization condition on these combinations.

Residual sequences consist of normalised combinations of Krylov sequence

Lemma 9 *Let a matrix A, a vector f, and a sequence X be given. Let $R = R\langle A,X,f\rangle$ and $K = K\langle A,x_0,f\rangle$. Then*

$$\exists_{\{\pi_i \in \mathbf{P}^{(\mathbf{n})}\}} : X = P\langle \{\pi_i\}_{i\geq 1}, A, M, x_0, f\rangle \tag{19}$$

$$\Leftrightarrow \quad x_{i+1} = x_1 + \pi_i(M^{-1}A)M^{-1}r_1 \tag{20}$$

$$\Leftrightarrow \quad \exists_{U\in\mathbf{U}} : X(J-E_1) = KU \tag{21}$$

$$\Leftrightarrow \quad \exists_{V\in\mathbf{U}^{(\mathbf{1})}} : R\langle A,X,f\rangle = K\langle AM^{-1}, x_0, f\rangle V \tag{22}$$

$$\Leftrightarrow \quad \exists_{\{\phi_i \in \mathbf{P}^{(\mathbf{n},\mathbf{1})}\}} : R\langle A,X,f\rangle = [\phi_i(AM^{-1})r_1] \tag{23}$$

where and the ϕ_i and π_i polynomials are related by

$$\phi_{i+1}(x) = 1 + x\pi_i(x), \tag{24}$$

U contains the coefficients of the π polynomials and V contains the coefficients of the ϕ polynomials, and

$$V = \begin{pmatrix} 1 & \cdots \\ 0 & U \end{pmatrix}.$$

Proof: Suppose X is generated by a polynomial iterative method $P\langle\{\pi_i\}_{i\geq 1}, A, M, x_0, f\rangle$, satisfying 21 by definition. Equation equation (22) is the matrix formulation of this statement.

Multiplying equation equation (21) by A and subtracting f on both sides gives

$$r_{i+1} = r_1 + AM^{-1}\pi_i(AM^{-1})r_1,$$

in other words, $r_{i+1} = \phi_{i+1}(AM^{-1})r_1$ with ϕ_i satisfying equation (24). This establishes equation (23).

We can also multiply equation (22) by A, giving

$$R(J-E_1) = AKU = KJU;$$

Noting that $RE_1 = KE_1$ we find

$$RJ = K(E_1 + JU).$$

Attaching r_1 to the left of this, we get

$$R = K(E_1 + JUJ^t)$$

where we oberve that JUJ^t has U as its $2:,2:$ subblock. By lemma 7 it follows that $R = KU$ where $U = U(\phi_i) \in \mathbf{U}^{(1)}$ and $K = K\langle AM^{-1}, r_1\rangle$. It is easy to see that all implications in this proof are equivalences. •

Block statement
In the block version of this lemma, U is a block upper triangular matrix with identity blocks in the first block row.
End of block statement

Remark 10 *Equation (23) states that the residuals are combinations of the Krylov space vectors. This should not be interpreted as a computational statement, that is, residuals are usually not constructed from computed Krylov vectors. Generating the Krylov space explicitly is essentially the 'power method'. Taking subsequent combinations of this sequence, for instance to orthogonalise it, is quickly numerically unstable.*

Remark 11 *Note that, even though we started with a left-preconditioned method, R consists of combinations of a Krylov sequence with coefficient matrix AM^{-1}, that is, a right-preconditioned matrix.*

Extended statement
The update equation $X(J-I) = WU$ gives the update statement $R(J-I) = AWU$ for R; however, it is not possible to go beyond that to a statement about R itself.
End of extended statement

We make some simple observations.

Iterative method can be updated with residuals

Corollary 12 *Let X be a preconditioned polynomial iterative method, and R its residuals, then there are upper triangular matrices U, V such that*

$$X(J-E_1) = RU, \quad X(J-I) = RV.$$

Proof: This follows from lemmas 8 and 9. •

Residuals satisfy update relation

Lemma 13 *Let R be a residual sequence, then there are polynomials $\{\pi_i\}$ such that*

$$r_{i+1} = r_i + AM^{-1}\pi_i(AM^{-1})r_1. \tag{25}$$

holds.

Proof: This immediately follows from corollary 12, but we also give a direct proof. By lemma 9 we know that $r_{i+1} = \tilde{\pi}_{i+1}(AM^{-1})r_1$, where $\tilde{\pi}_{i+1}$ is a polynomial normalised to $\tilde{\pi}_{i+1}(0) = 1$. Thus there are polynomials $\{\pi_i\}$ such that

$$AM^{-1}\pi_i(AM^{-1}) = \tilde{\pi}_{i+1}(AM^{-1}) - \tilde{\pi}_i(AM^{-1}).$$

This gives the desired result. •

This has the following practical implication.

Remark 14 *If we derive an update relation for residuals, the corresponding update relation for the iterates can be found by 'dividing out A'. The update relation for the iterates then has the expected form*

$$x_{i+1} = x_i + \pi_i(M^{-1}A)M^{-1}r_1$$

in which we recognise the left-preconditioned form.

Corollary 15 *Let R be a preconditioned residual sequence, then*

$$r_{n+1} - r_1 = AM^{-1}R_n v_n \tag{26}$$

for some vector v_n, or in block statement

$$R_{n+1}(J - E_1) = AM^{-1}R_n V \tag{27}$$

with V upper triangular.

Proof: From equation (23) we get

$$r_{n+1} - r_1 = \sum_{i=1}^{n+1}(AM^{-1})^i r_1 = AM^{-1}\sum_{i=0}^{n}(AM^{-1})^i r_1 = AM^{-1}Rv_n$$

for some vector v_n. •

By lemma 5 we can also write equation (27) as

$$R(J - I) = AM^{-1}RV \tag{28}$$

with V a different upper triangular matrix.

Remark 16 *Writing equation (27) as $AM^{-1}R = R(J - E_1)V^{-1}$, we obtain a relation involving A, M, R, and a upper Hessenberg matrix. We will derive the same relation differently in section 4, equation (31), and study this relation in much more detail, paying particular attention to the fact that this Hessenberg matrix has zero column sums.*

Extended statement
Extending this remark, we find

$$AM^{-1}W = RH \qquad \text{or} \qquad AW = RH$$

for the two forms of extended iterative methods, with H of upper Hessenberg form and with zero column sums.
End of extended statement

Question: *Is this R the residual sequence of X? Prove general statement.*

Here is the matrix formulation of lemma 9.

Upper triangular matrix for X is subblock of upper triangular matrix for R

Lemma 17 *A polynomial iterative method X and its residual sequence R satisfy*

$$X_{n+1}(J - E) = KU_n, \quad R_{n+1} = K_{n+1}V_{n+1} \qquad \text{where } K = K\langle M^{-1}A, M^{-1}r_1\rangle,$$

•

with the upper triangular matrices U and V related by

$$V_{n+1} = E_1 + JU_nJ^t, \qquad \textit{or, more pictorily,} \qquad V_{n+1} = \begin{pmatrix} 1 & e^t \\ \emptyset & U_n \end{pmatrix}.$$

Proof: Let $X_{n+1}(J-E) = KU_n$ be given. Then, from $M^{-1}RE = KE$:

$$\begin{aligned} M^{-1}R_{n+1}(J-E) &= M^{-1}AK_nU_n = K_{n+1}JU_n \\ M^{-1}R_{n+1}J &= K_{n+1}E + K_{n+1}JU_n \\ M^{-1}R_{n+1}JJ^t &= K_{n+1}EJ^t + JUJ^t \end{aligned}$$

Now observe that JJ^t is I except for its $(1,1)$ element, that the first column of the rhs is zero, that $M^{-1}r_1 = k_1$, and that KEJ^t is KE except for its $(1,1)$ element. In sum, adding a first column of $M^{-1}r_1$ to both sides gives $M^{-1}R = K(E+JUJ^t)$. All implications in this proof can be reversed. •

Question: *Can we also show something like this for $X(J-I) = RU$?*

3.5.1 Affine combinations of residual sequences

We will occasionally consider sequences that are formed by taking certain combinations of a residual sequence where the coefficients sum up to 1, i.e.,

$$g_j = \sum_{i \le j} r_i u_{ij}, \qquad \sum_{i \le j} u_{ij} = 1. \tag{29}$$

Definition: Affine combinations

Definition 9 *The weighted sum*

$$x = \sum_i \alpha_i y_i, \qquad \sum_i \alpha_i = 1$$

is called an affine combination[56].

Affine combinations of a residual sequence are a residual sequence

Lemma 18 *Let R and G be sequences related by $G = RU$ where U is a non-singular upper triangular matrix with column sums $\equiv 1$, that is,*

$$e^tU = e^t.$$

Then R is a residual sequence iff G is a residual sequence.

Proof: Noting that $e^tU = e^t$ iff $e^tU^{-1} = e^t$ we only prove the implication one way. Let R be a residual sequence, i.e., by lemma 9 $R = KV$ with K a Krylov sequence and V a non-singular upper triangular matrix in $\mathbf{U^{(1)}}$, that is, with $v_{1j} \equiv 1$. Then $G = RU = KVU$ and VU satisfies the same properties as V, so, again by lemma 9, G is also a residual sequence. •

Block statement

For the block version of this lemma, where each u_{ij} is a $k \times k$ matrix, we recall that matrices U in $\mathbf{U^{(1)}}$ have blocks $u_{1j} \equiv I_{k\times k}$. For the lemma to hold true, we have to interpret the column sums being $\equiv 1$ as

$$(e^t \otimes I_k) \times U = e^t \otimes I_k \qquad \text{that is,} \qquad \forall_j \colon \sum_i u_{ij} = I_k, \tag{30}$$

which is a stronger statement than the column sums being $\equiv 1$ in a scalar sense. The rest of the statement and

5. Let V be a linear subspace and $a \notin V$; let $y_i \in a+V$ and $\sum_i \alpha_i = 1$, then $x = \sum_i \alpha_i y_i \in a+V$.

6. The better known, similar, definition for *convex* combinations involves the extra condition that all $\alpha_i \ge 0$

proof proceed analogously.
End of block statement

We can also get a result that is in a way the reverse statement:

All residual sequences from r_1 are affine combinations of each other

Lemma 19 *Let R and $\bar{R}$ be residual sequences wrt the same A and f, and $r_1 = \bar{r}_1$, then there is an upper triangular matrix V with column sums $\equiv 1$ such that $\bar{R} = RV$.*

Proof: Since $r_1 = \bar{r}_1$, R and $\bar{R}$ are combinations of the same Krylov sequence K, say $R = KU$ and $\bar{R} = K\bar{U}$ with both $U, \bar{U} \in \mathbf{U}^{(1)}$. This gives $\bar{R} = RV$ with $V = U^{-1}\bar{U}$. From $\bar{U} = UV$ we get for all n

$$1 = \bar{u}_{1n} = \sum_j u_{1j} v_{jn} = \sum_j v_{jn},$$

that is, V has column sums $\equiv 1$, which with $\bar{R} = RV$ shows that $\bar{R}$ consists of affine combinations of R. •

This lemma will be used in section 12.7.

We can extend the update relation of equation (28) to let residual sequences that are combinations of each other be updated in terms of each other.

Residual sequences can be updated with each other

Lemma 20 *Let R and G be residual sequences with $r_1 = g_1$. Then G satisfies an update relation*

$$G_{n+1}(J-I) = AM^{-1}R_nV$$

with V upper triangular.

Proof: Let V_n by upper triangular such that $R_{n+1}(J-I) = AM^{-1}R_nV_n$ (which exists, see equation (28)), and $G_{n+1} = R_{n+1}U_{n+1}$ with $e^tU_{n+1} = e^t$ (the existence of this U_{n+1} follows from lemma 19). Then

$$\begin{aligned} G_{n+1}(J-I) &= R_{n+1}U_{n+1}(J-I) \\ &= R_{n+1}(J-I)\tilde{U}_n \\ &= AM^{-1}R_nV_n\tilde{U}_n \\ &= AM^{-1}R_n\tilde{V}_n \end{aligned}$$

where the existence of $\tilde{U}_n$ follows from lemma 26, and $\tilde{V}_n = V_n\tilde{U}_n$, about which we note that $\tilde{U}_ne = e$ and thus $V_ne = \tilde{V}e$. •

Question: *We need to be slightly more careful about square/rectangular matrices in the above.*

3.6 Termination of the iterative process

In this section we investigate the conditions on termination (in exact arithmetic) of polynomial iterative methods.

Theorem 21 *Let X be a polynomial method, let $K = K\langle A, x_0, f\rangle$, and let n be such that K_n is an A-invariant subspace. It is possible to choose π_n such that $x_n = x_1 + \pi_n(A)f = A^{-1}f$.*

Proof: If K_n is A-invariant, $k_{n+1} \in [K_n]$, in other words

$$K_{n+1}c_{n+1} = 0 \qquad \text{some vector } c_{n+1} \in \mathbf{R}^{n+1},$$

and we can assume without loss of generality that $c_{n+1,1} = 1$. Let ϕ_{n+1} be the polynomial with coefficients in c_{n+1}, then $\phi_{n+1}(A)r_1 = 0$, so, with π_n related to ϕ_{n+1} by equation (24), we find by theorem 2 that $x_n = A^{-1}f$. •

The most interesting polynomial methods are those that orthogonalise the R sequence under some inner product.

Corollary 22 *Let the residual sequence R be orthogonal under some inner product, and let n be the smallest index for which $r_{n+1} = 0$. Then $x_n = A^{-1}f$.*

Proof: If K_n is an independent set, $R_n = K_n U_n$ can be chosen as nonzero orthogonal vectors. The occurrence of $r_{n+1} = 0$ then clearly marks an invariant subspace K_n, which by the above theorem implies that the solution has been reached. •

Question: *Something missing here. What condition guarantees that r_i is nonzero as long as it is possible?*

Question: *How is terminations of block sequences done? What happens if some coefficient matrix merely becomes singular, not zero?*

4 Hessenberg matrices

Previous section: 3 Next section: 5

The subject of Hessenberg matrices comes up often in the discussion of polynomial iterative methods. If we combine the equations for a Krylov sequence, $AK = KJ$, and for residuals as combinations of the Krylov sequence, $R = KU$, with U upper triangular, we find that

$$AR_n = R_{n+1}H, \qquad \text{with } H = U^{-1}JU$$

where clearly H is of upper Hessenberg form. Recall that in remark 16 we saw a different origin of Hessenberg matrices.

In the preconditioned case we have $R = KU$ where $K = K\langle AM^{-1}, k_1\rangle$ (see remark 11), leading to

$$AM^{-1}R = RH, \qquad \text{with } H = U^{-1}JU. \tag{31}$$

We will state more about the relation between the residual sequence and the Hessenberg matrix in section 4.3; in particular we shall see (lemma 29) that the connection between combinations of Krylov sequences and Hessenberg matrices is an equivalence. However, in this section we start out by proving a number of facts about Hessenberg matrices independent of any connection to iterative methods.

4.1 Hessenberg matrices with zero column sums

Combining the fact that residuals sequences are normalised combinations (lemma 9) $R = KU$ with $u_{1j} \equiv 1$ of a Krylov sequence, and that the Hessenberg matrix relating the residuals to the system by $AM^{-1}R = RH$ equals $H = U^{-1}JU$, we derive an important property of H: it has zero column sums. We will then state some facts about such Hessenberg matrices.

H zero column sums iff U normalized

Lemma 23 *Let U be a nonsingular upper triangular matrix, and let $H = U^{-1}JU$. The Hessenberg matrix H has zero column sums iff the first row of U is constant.*

Proof: With the zero vector and the all-ones vector e we can formulate the zero column sums as $e^tH = 0^t$. Then

$$\begin{aligned} e^tH = e^tU_{n+1}^{-1}JU_n = 0^t \quad &\Leftrightarrow \quad e^tU_{n+1}^{-1}J = 0^t \\ &\Leftrightarrow \quad e^tU_{n+1}^{-1} = (\alpha,0,0,\ldots) \qquad \text{some nonzero } \alpha \\ &\Leftrightarrow \quad \alpha^{-1}e^t = (1,0,0,\ldots)U_{n+1} \end{aligned}$$

(in the transition from the first to the second line we use the fact that U is nonsingular to state that α is nonzero) which proves the statement. •

Block statement

The analogous block statement is sufficiently different that we give it its own lemma.

Lemma 24 *Let U be a nonsingular block upper triangular matrix with nonsingular diagonal blocks, and let $H = U^{-1}JU$. The first block row of U consists of identical elements iff (in a block sense)*

$$\forall_j\colon \sum_i h_{ij} = \emptyset. \tag{32}$$

Proof: The proof is as above, but with α being any nonsingular matrix. •

End of block statement

Next we consider the LU factorisation of a Hessenberg matrix with zero column sums.

$H = (I-J)U$ iff zero column sums

Lemma 25 *Let H be an upper Hessenberg matrix of maximum rank, then H can be factored as $H = (I-J)U$ iff*

$$\begin{cases} H \text{ is of size } (n+1)\times n \text{ and all its columns sums are zero, or} \\ H \text{ is of size } n\times n \text{ and all its columns sums except for the last are zero;} \end{cases}$$

the $I-J$ factor is of the same shape as H and U is square with dimension the column dimension of H.

Proof: Trivially, if $H = (I-J)U$ its column sums are zero if it is rectangular and all but the last zero if it is square, since $e^t(I-J)_{[n+1,n]} = 0^t$. Conversely, the first column sum being zero imply that $h_{21} = -h_{11}$. Since the matrix has maximum rank, the elements in the first column are nonzero. The first elimination step of Gaussian elimination then entails adding the first row to the second. This implies that the first column of the L factor can be chosen as $(1,-1,0,0,\ldots)^t$.

Since elimination did not involve scaling the pivot row, there was no change to the column sums of any column, nor did the elimination change the (column) rank of the remaining block. Hence the remaining $2:*,2:*$ block of H is again a Hessenberg matrix of full rank with properties as stipulated, and we can inductively repeat this argument. •

Block statement
This lemma holds true in the block case, if we substitute $J \to J \otimes I_k$, and we interpret the zero column sums in the stronger sense of equation (32). The matrix U is block upper triangular.
End of block statement

Sometimes we are interested in a factorisation of the form $C(J-I)$ rather than $(J-I)B$.

$U(I-J) = (I-J)V$ if U constant column sums, **this only works for H square**

Lemma 26 *Let U_{n+1} be an upper triangular matrix with constant column sums α. Then $H := U_{n+1}(I-J)$ can be written as $H = (I-J)V_n$, where V_n is upper triangular with constant row sums α.*

Proof: Let U_{n+1} have constant column sums α, that is, $e^tU = \alpha e^t$. Then $H := U_{n+1}(I-J)$ satisfies $e^tH = 0^t$, so from lemma 25 H can be written as $H = (I-J)V_n$ with V_n an upper triangular matrix. From $He = U_{n+1}(1,0,\ldots,0,-1)^t$ and $He = (I-J)V_ne$ we find for $f = V_ne$ that $(I-J)f = u_{11}e$, so $Ve = u_{11}e$. •

4.2 QR decompositions of Hessenberg matrices

For further reading: the material in this section is not referred to elsewhere.

The *QR* decomposition of the Hessenberg matrix with zero column sums takes a remarkably simple form.

Lemma 27 *Let H be an upper Hessenberg matrix with zero column sums, and let $H = QR$ be a decomposition into an orthonormal matrix and an upper triangular matrix. Then Q is given by*

$$q_{kn} = -\frac{1}{\sqrt{n(n+1)}} \quad k \le n; \qquad q_{n+1n} = \sqrt{\frac{n}{n+1}}.$$

Furthermore, $Q = (J-I)B^{-1}$, where B is an upper bidiagonal matrix.

Proof: H has zero column sums, so Q has zero column sums. The values given satisfy this requirement plus orthonormality of the columns. Then, with $\alpha_n = \sqrt{n(n+1)}$:

$$\begin{aligned} Q &= \begin{pmatrix} -\alpha_1^{-1} & -\alpha_2^{-1} & \\ \alpha_1^{-1} & -\alpha_2^{-1} & \\ & 2\alpha_2^{-1} & \cdots \\ & & \ddots \end{pmatrix} \\ &= \begin{pmatrix} -1 & & & \\ 1 & -1 & & \\ & 1 & -1 & \\ & & \ddots & \ddots \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & \cdots \\ & 2 & 2 & \cdots \\ & & 3 & \cdots \\ & & & \ddots \end{pmatrix} \mathrm{diag}(\alpha_i^{-1}) \\ &= (J-I)\left[\mathrm{diag}(\alpha_i)(I-J^t)\mathrm{diag}(i^{-1})\right]^{-1} \end{aligned}$$

that is, $Q = (J-I)B^{-1}$ with

$$B = \begin{pmatrix} \sqrt{\frac{2}{1}} & -\sqrt{\frac{1}{2}} & & \\ & \sqrt{\frac{3}{2}} & -\sqrt{\frac{2}{3}} & \\ & & \ddots & \ddots \end{pmatrix}.$$

•

We are sometimes interested in relations between the *QR* decompositions of Hessenberg matrices that are equivalent through diagonal transformations.

QR decomps of equiv Hess related by two bidiagonal mats

Lemma 28 *Let $H_1 = Q_1U_1$ and $H_2 = Q_2U_2$ be QR decompositions of Hessenberg matrices that are related by $H_1 = \Omega H_2 \Omega^{-1}$ where Ω is a diagonal matrix. Then there is an upper triangular matrix T such that*

$$Q_2 = \Omega^{-1}Q_1T, \qquad U_2 = T^{-1}U_1\Omega.$$

If H_2 has zero column sums, T takes the form $B_1B_2^{-1}$ where B_1 and B_2 are upper bi-diagonal matrices.

Proof: We have $H_1\Omega = Q_1U_1\Omega = \Omega H_2 = \Omega Q_2U_2$, so $Q_1^t\Omega Q_2 = U_1\Omega U_2^{-1} \equiv T$, and T is clearly upper triangular. This proves the first statement of the lemma.

If H_2 has zero column sums, by lemma 27 its *QR* decomposition satisfies $Q_2 = (J-I)B_2^{-1}$, where B_2 is upper bi-diagonal. Since $Q_1^t\Omega Q_2$ is upper triangular, $Q_1^t\Omega(J-I)$ is also upper triangular, but it is also of lower Hessenberg form, hence it is of upper bi-diagonal form, say

$$B_1 \equiv Q_1^t\Omega(J-I). \tag{33}$$

From

$$Q_1^t \Omega (J-I) = U_1 \Omega U_2^{-1} B_2$$

we find that $TB_2 = U_1 \Omega U_2^{-1} B_2 = B_1$. This proves the second statement of the lemma. •

For future reference we note that

$$B_1 \equiv Q_1^t \Omega (J-I) = TB_2 = U_1 \Omega U_2^{-1} B_2,$$

we have

$$H_2 \Omega^{-1} U_1^{-1} = Q_2 U_2 \Omega^{-1} U_1^{-1} = Q_2 B_2 B_1^{-1} = (J-I) B_1^{-1}. \qquad (34)$$

4.3 Hessenberg matrices of residual sequences

The following auxiliary lemma states the connection between Hessenberg matrices and Krylov sequences.

In $AM^{-1}R = RH$, H Hessenberg iff R is combination of Krylov sequence

Lemma 29 *Let $AM^{-1}R = RH$, let $K = K\langle AM^{-1}, k_1\rangle$, and let $r_1 = k_1\alpha$ for some $\alpha \neq 0$, then H is an irreducible upper Hessenberg matrix iff there is a nonsingular upper triangular matrix U such that $R = KU$; U and H are related by $H = U^{-1}JU$.*

Proof: If U is a nonsingular upper triangular matrix, $AM^{-1}K = KJ$, and $R = KU$, then

$$AM^{-1}R = RU^{-1}JU$$

where $U^{-1}JU$ is of irreducible upper Hessenberg shape.

Conversely, if H is an upper Hessenberg matrix, $r_1 = k_1\alpha$ for some $\alpha \neq 0$, and $AM^{-1}R_{n-1} = R_n H$, then an upper triangular matrix U_n can be determined such that $R_n = K_n U_n$, namely U has to satisfy

$$U_n H = JU_{n-1}$$

and this can be solved recursively. For instance, noting that the first row of JU_{n-1} is zero (and picking $u_{11} = \alpha$):

$$\begin{aligned} u_{11}h_{11} + u_{12}h_{21} = 0 \quad &\Rightarrow \quad u_{12} = -\alpha h_{11} h_{21}^{-1}, \\ u_{11}h_{12} + u_{12}h_{22} + u_{13}h_{32} = 0 \quad &\Rightarrow \quad u_{13} = -(\alpha h_{11} + u_{12}h_{22}) h_{32}^{-1}, \\ u_{11}h_{13} + \cdots = 0 \quad &\Rightarrow \quad u_{14} = -(\ldots) h_{43}^{-1}, \quad \text{et cetera} \end{aligned}$$

Then for the second row

$$\begin{aligned} u_{11} = u_{22}h_{21} \quad &\Rightarrow \quad u_{22} = (\ldots) h_{21}^{-1} \\ u_{12} = u_{22}h_{22} + u_{23}h_{32} \quad &\Rightarrow \quad u_{23} = (\ldots) h_{32}^{-1}, \quad \text{et cetera} \end{aligned}$$

and in general

$$\forall_i \forall_{j \geq i-1} : u_{ij+1} h_{j+1j} = u_{i-1j} - \sum_{i \leq k \leq j} u_{ik} h_{kj}.$$

We see that U can be solved if all $h_{i+1,i} \neq 0$, that is, if H is irreducible. Now

$$AM^{-1}\, RU^{-1} = RHU^{-1} RU^{-1}(UHU^{-1}) = RU^{-1}J$$

so $RU^{-1} = K\langle AM^{-1}, r_1 u_{11}^{-1} \rangle$. Since we choose u_{11} to satisfy $r_1 u_{11}^{-1} = k_1$, we find $RU^{-1} = K$, that is, $R = KU$. •

Block statement
A block extension of the above statement holds, if we define a block Hessenberg matrix to be irreducible if the lower diagonal blocks are nonsingular.
End of block statement

We can now collect some results to state the nature of the Hessenberg matrix associated with a residual sequence.

In $AM^{-1}R = RH$, H zero column sum Hessenberg iff R residual sequence

Theorem 30 *Let a matrix A, a preconditioner M, a sequence R, and an irreducible upper Hessenberg matrix H related by $AM^{-1}R = RH$ be given, then R is a residual sequence if and only if H has zero column sums.*

Proof: Combine lemmas 9, 29 and 23. •

Block statement
In the block case, R is a block residual sequence iff in $AR = RH$ the block Hessenberg matrix H has zero column sums in the strong sense of equation (32).
End of block statement

In lemma 18 we saw that affine combinations of residual sequences form again a residual sequence. We can give a corresponding statement in terms of Hessenberg matrices.

Lemma 31 *Let R be a residual sequence, and G a series of affine combinations of R, then $AG = GH$ with H a Hessenberg matrix with zero column sums.*

Proof: Trivial. •

5 Characterisation of polynomial iterative methods

Previous section: 4 Next section: 6

We now summarise the preceding sections by giving a number of equivalent formulations for the preconditioned solution of a linear system.

Theorem 32 *Let a vector $\bar{x}$ and a sequence X be given, and let A and f be such that $A\bar{x} = f$. Define residuals by $R = AX - fe^t$, then the following statements are equivalent:*

1. *The sequence X is a preconditioned polynomial method for $\bar{x} = A^{-1}f$.*
2. *There is a nonsingular matrix M such that the sequence X satisfies*
$$x_{i+1} - x_1 \in [M^{-1}r_1, \ldots, M^{-1}(AM^{-1})^{i-1}r_1]. \tag{35}$$
3. *There are a nonsingular matrix M and upper triangular matrix U such that*
$$X(J-I) = M^{-1}RU,$$
that is,
$$x_{i+1} - x_i = \sum_{j\le i} M^{-1}r_j u_{ji}. \tag{36}$$
4. *There are a nonsingular matrix M and Hessenberg matrix H with zero column sums such that*
$$AM^{-1}R = RH$$
that is,
$$AM^{-1}r_i = h_{i+1,i}r_{i+1} + \cdots + h_{1i}r_1. \tag{37}$$
5. *There are a sequence P (the 'search directions'), a nonsingular matrix M, a diagonal matrix D, and an upper triangular matrix U with unit diagonal such that*
$$APD = R(I-J), \qquad PU = M^{-1}R$$
that is,
$$r_{i+1} = r_i - Ap_i d_{ii}, \qquad p_i = M^{-1}r_i - \sum_{j<i} p_j u_{ji}. \tag{38}$$
6. *There are a nonsingular matrix M and upper triangular matrix $U \in \mathbf{U}^{(1)}$ such that*
$$R = KU, \qquad K = K\langle AM^{-1}, r_1\rangle.$$
7. *There are a nonsingular matrix M and polynomials $\{\pi_i\}_{i\ge 1}$ such that*
$$r_i = \pi_i(AM^{-1})r_1, \qquad \deg(\pi_i) = i-1, \qquad \pi_i(0) = 1.$$

Proof: We prove the equivalence of the statements in sequence.

$1 \Leftrightarrow 2$ Statement 2 is a simple rephrasing of the definition of a preconditioned polynomial method.

$2 \Leftrightarrow 6$ *and* 7 Multiplying equation (35) by A, we find that
$$r_{i+1} - r_1 \in [AM^{-1}r_1, \ldots, (AM^{-1})^i r_1].$$
Introducing the Krylov sequence K satisfying
$$k_1 = r_1, \qquad AM^{-1}K = KJ,$$
we find that $R = K\tilde{U}$ where $\tilde{U}$ is an upper triangular matrix with $\tilde{u}_{1*} \equiv 1$. This shows the equivalence with statements 6 and 7.

$2 \Leftrightarrow 3$ Since statement 2 can be expressed as $X(J-I) = M^{-1}K\bar{U}$ with U upper triangular, we find statement 3 where $U = \tilde{U}\bar{U}$. For the reverse implication, note that a nonsingular upper triangular matrix U can easily be split as a product $\tilde{U}\bar{U}$ where $\tilde{u}_{1*} \equiv 1$.

$3 \Leftrightarrow 4$ We now find statement 4 by multiplying the previous statement by A, and defining $H = (J-I)U^{-1}$. For the reverse implication, lemma 25 tells us that a Hessenberg matrix with zero column sums can indeed be split on the form $(J-I)U^{-1}$.

$4 \Leftrightarrow 5$ Statement 5 follows by splitting the previous statement.

Many of the clauses of the above theorem look familiar from the literature on Conjugate Gradient-like methods. However, we stress that this theorem is general: it covers all sorts of iterative methods, whether based on conjugacy or not.

Various different looking variants of Krylov method exist that are still equivalent.

Left-preconditioned residual equation

Corollary 33 *The equations*

$$X(J-I) = RU, \qquad M^{-1}AR = RH \tag{39}$$

define a polynomial iterative method if $r_1 = M^{-1}(Ax_1 - b)$.

Proof: Start with a standard formulation

$$X(J-I) = M^{-1}RU, \qquad AM^{-1}R = RH,$$

split $U = U_1U_2$, and write $\tilde{R} = M^{-1}RU_1$, then

$$X(J-I) = \tilde{R}U_2, \qquad M^{-1}A\tilde{R} = \tilde{R}\tilde{H}$$

with $\tilde{H} = U_1^{-1}HU_1$.

6 Minimization

Previous section: 5 Next section: 7

We will prove some minimisation results for Krylov methods; in particular we will investigate the relation between minimization and orthogonalisation. Constructive aspects of orthogonalisation are discussed in section 7. Also, in this section we only prove that a minimum norm is attained; the actual value of the minimum, that is, the speed of convergence of the method, will be discussed in section 14.

First of all we observe the equivalence of error and residual minimisation under norms induced by the coefficient matrix. In the case of symmetry we have a family of equivalence statements:

$$\|e\|_{A^k}^2 = (x-\bar{x})^tA^k(x-\bar{x}) = (Ax-f)^tA^{k-2}(Ax-f) = \|r\|_{A^{k-2}}^2.$$

In the nonsymmetric case we only have

$$\|r\|_{A^{-1}}^2 = (Ax-f)^tA^{-1}(Ax-f) = (x-\bar{x})^tA^t(x-\bar{x}) = \|e\|_A^2.$$

6.1 Minimization of residuals in L_2 norm

The following very general theorem can be found in [30], also cited in [68].

Theorem 34 *Let a matrix A, vectors x_1 and f, and space K_m be given. The vector $\bar{x}$ is a solution of the Petrov-Galerkin problem*

$$\textit{find } x \in x_1 + K_m \textit{ such that } (Ax-f,v) = 0 \textit{ for all } v \in AK_m$$

if and only if $\bar{x}$ uniquely minimises $Ax-f$ over the space x_1+K_m.

Proof: Let $\bar{x} \in x_1 + K_m$ solve the Petrov-Galerkin problem, and let $x \in x_1 + K_m$, then

$$\begin{aligned} \|Ax-f\|^2 &= \|A[(x-\bar{x})+\bar{x}]-f\|^2 = \|A(x-\bar{x})+(A\bar{x}-f)\|^2 \\ &= \|A\bar{x}-f\|^2 + 2\big(A(x-\bar{x}),A\bar{x}-f\big) + \|A(x-\bar{x})\|^2 \end{aligned} \tag{40}$$

where the middle term in the last line drops out by the Petrov-Galerkin condition. Hence

$$\|Ax-f\|^2 \geq \|A\bar{x}-f\|^2$$

with equality only if $x=\bar{x}$.

Conversely, suppose $\bar{x} \in x_1+K_m$ uniquely minimises $\|Ax-f\|$. Then obviously, for any z,

$$Q(\alpha) = \|A\bar{x}-f+\alpha Az\|^2$$

is minimised for $\alpha=0$. Differentiating Q and substituting $\alpha=0$ gives $(Az, A\bar{x}-f)=0$, which is the Petrov-Galerkin condition. •

Block statement

For a block version of theorem 34 over a space of block-vectors of width s, we need to interpret x_1+K_m as

$$[x_1^{(1)}+K_m, \ldots x_1^{(s)}+K_m].$$

Furthermore, we need to define the norm of $N \times s$ block vectors as

$$\|x\|_{\mathbf{R}^N} = \|x^t x\|_{R^s}.$$

End of block statement

Remark 35 *If we would want to formulate the above theorem for more general inner products and norms, say induced by a positive definite matrix B, the middle terms in equation (40) becomes*

$$(A(x-\bar{x}), A\bar{x}-f)_B + (A(x-\bar{x}), A\bar{x}-f)_{B^t}.$$

Hence, we have to impose that B is symmetric *positive definite. In particular, we can not use powers of A here, although powers of A^tA or AA^t are feasible.*

Theorem 34 is related to projections. (See the basic theory of projections in section 7.7.) The statement that $x \in x_1+K_m$ is equivalent to $r \in r_1+AK_m$. Minimising r then means minimising the difference between $u_m \in AK_m$ and $-r_1$: $r_1+u_m = u_m-(-r_1)$. (Note that if we could actually find $u_m=-r_1$, then the new residual would be zero and we would have found the solution of the system.) The theorem now states that the shortest distance between $-r_1$ and $u_m \in AK_m$ is that choice of u_m for which the difference vector $r=r_1+u_m$ is orthogonal to AK_m, that is, we project $-r_1$ onto AK_m.

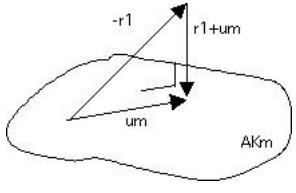

Figure 1: The optimal update r_m make the new residual orthogonal to the AK_m subspace

We can apply theorem 34 to polynomial iterative methods in a straightforward manner.

Minimisation of residuals $\Leftrightarrow$ AM^{-1}-orthogonal residuals

Corollary 36 *Let X be a polynomial iterative method for the system $Ax=f$, with R be the residual sequence $R\langle A,X,f\rangle$, and suppose the residuals are linearly independent. Then r_{n+1} is of minimal length over all polynomials in the n-th step of the iterative method, if and only if r_{n+1} is AM^{-1}-orthogonal to all $r_1,\ldots,r_n$.*

Proof: Every iterate in a polynomial method can be written as $x_{n+1}=x_1+M^{-1}R_nu_n$, so we are in the situation for which theorem 34 applies, with $K_m=$

$M^{-1}R_m$. By this theorem, r_{n+1} is minimised iff it is orthogonal to AK_n, in other words, AM^{-1}-orthogonal to R_m. •

We will also give a direct proof, expressed solely in terms of the residuals sequence.

AM^{-1}-orthogonal R is minimal

Theorem 37 *Let R be a residual sequence satisfying $AM^{-1}R = RH$, and let $R^tAM^{-1}R$ be upper triangular, then the residuals are of minimum length in the sense that*

- *if $\tilde{R}$ is a residual sequence satisfying $AM^{-1}\tilde{R} = \tilde{R}\tilde{H}$, and $r_1 = \tilde{r}_1$,*
- *then $\|r_i\| \leq \|\tilde{r}_i\|$ for all i.*

Proof: Let the Krylov sequence $K = K\langle AM^{-1}, r_1\rangle$, and let

$$R = KU, \quad \tilde{R} = K\tilde{U} \qquad \text{where } U, \tilde{U} \in \mathbf{U}^{(\mathbf{1})}$$

be residual sequences. Then $AM^{-1}R = RH$ and $AM^{-1}\tilde{R} = \tilde{R}\tilde{H}$, where by theorem 30, lemma 25, and lemma 5 the Hessenberg matrices can be factored

$$H = (J - E_1)V, \quad \tilde{H} = (J - E_1)\tilde{V} \qquad \text{with } V, \tilde{V} \text{ upper triangular.}$$

Use the fact that $RE_1 = \tilde{R}E_1$ to combine this as

$$\begin{array}{l} AM^{-1}KUV^{-1} = R(J - E_1) \\ AM^{-1}K\tilde{U}\tilde{V}^{-1} = \tilde{R}(J - E_1) \end{array} \quad \Rightarrow \quad \tilde{R}J = RJ + AM^{-1}KW,$$

with $W = \tilde{U}\tilde{V}^{-1} - UV^{-1}$. It now follows that

$$\begin{aligned} & J^t\tilde{R}^t\tilde{R}J \\ = \; & J^tR^tRJ + J^tR^tAM^{-1}KW + W^tK^tM^{-t}A^tRJ \\ & + W^tK^tM^{-t}A^tAM^{-1}KW. \end{aligned}$$

Now, the second and third term are strictly upper and lower triangular because of the upper triangularity of $R^tAM^{-1}K = R^tAM^{-1}RU^{-1}$, and the last term is of the form Y^tY, hence having a positive diagonal. Thus $\tilde{r}_i^t\tilde{r}_i \geq r_i^tr_i$ with equality only if $W = 0$, that is if $\tilde{R} = R$. •

Remark 38 *R being AM^{-1}-orthogonal in the above discussion can be stated in matrix terms as $R^tAM^{-1}R$ being upper triangular. (We used this as an explicit condition in theorem 37.) This will be the basis for derivations in section 8.2.1. In the fact that $R^t(AM^{-1})^tR$ is lower triangular we recognise the semi-orthogonality condition equation (42), with the choice $N^{-1} = AM^{-1}$ which we analyse in section 7.3.3.*

6.2 Minimization of residuals in the A^{-1} norm

We first give the general Petrov-Galerkin theorem [51], also cited in [68].

Theorem 39 *Let a symmetric positive definite matrix A, vectors x_1 and f, and space K_m be given. The vector $\bar{x}$ is a solution of the Petrov-Galerkin problem*

$$\text{find } x = x_1 + z, \text{ where } z \in K_m, \text{ such that}$$
$$(f - Ax, v) = 0 \text{ for all } v \in K_m$$

if and only if $\bar{x}$ minimises $Ax - f$ in the $\|\cdot\|_{A^{-1}}$ norm over the space $x_1 + K_m$.

Proof: Let $\bar{x} = x_1 + \bar{z}$ with $\bar{z} \in K_m$ solve the Petrov-Galerkin problem, and let $x = x_1 + z$ with $z \in K_m$, then

$$\begin{aligned} \|Ax - f\|^2_{A^{-1}} &= \|(A\bar{x} - f) + A(x - \bar{x})\|^2_{A^{-1}} \\ &= \|A\bar{x} - f\|^2_{A^{-1}} \\ &\quad -(x - \bar{x})^t A^t A^{-1}(A\bar{x} - f) - (A\bar{x} - f)^t A^{-1} A(x - \bar{x}) \\ &\quad +[A(x - \bar{x})]^t A^{-1}[A(x - \bar{x})], \end{aligned}$$

where the middle terms in the last line drop out by the Petrov-Galerkin condition, and using symmetry to simplify $A^t A^{-1} = I$. Using definitess of A we conclude

$$\|Ax - f\|^2_{A^{-1}} \geq \|A\bar{x} - f\|^2_{A^{-1}}.$$

Conversely, suppose $\bar{x} = x_1 + \bar{z}$ minimises $\|Ax - f\|_{A^{-1}}$ over $x_1 + K_m$. Then obviously, for any z,

$$Q(\alpha) = \|A\bar{x} - f + \alpha Az\|^2_{A^{-1}}$$

is minimised for $\alpha = 0$. Differentiating Q and substituting $\alpha = 0$ gives $z^t(A\bar{x} - f) = 0$, which is the Petrov-Galerking condition. •

The generalisation of this theorem to inner products induced by power of A is immediate:

Theorem 40 *Let a symmetric positive definite matrix A, an integer k, vectors x_1 and f, and space K_m be given. The vector $\bar{x}$ is a solution of the Petrov-Galerkin problem*

$$\text{find } x = x_1 + z, \text{ where } z \in K_m, \text{ such that}$$
$$(f - Ax, v)_{A^k} = 0 \text{ for all } v \in K_m$$

if and only if $\bar{x}$ minimises $Ax - f$ in the $\|\cdot\|_{A^{k-1}}$ norm over the space $x_1 + K_m$.

We will apply this theorem the case of polynomial iterative methods.

For A spd, M^{-1}-orthogonal R is minimal in A^{-1} norm

Corollary 41 *Let A be symmetric positive definite, let R be a residual sequence satisfying $AM^{-1}R = RH$, and let $R^t M^{-1} R$ be diagonal, then the residuals are of minimum length in the A^{-1}-norm in the sense that*

- *if $\tilde{R}$ is a residual sequence satisfying $AM^{-1}\tilde{R} = \tilde{R}\tilde{H}$, and $r_1 = \tilde{r}_1$,*
- *then $\|r_i\|_{A^{-1}} \leq \|\tilde{r}_i\|_{A^{-1}}$ for all i.*

Proof: This follows immediately from an application of theorem 39 by noting that iterates satisfy

$$x_{m+1} = x_1 + M^{-1}R_m u_m,$$

whence we define $K_m = [M^{-1}R_m]$, and that

$$(Ax_{m+1} - f)^t M^{-1} R_m = 0.$$

We will also give a direct proof of the statement. Let the Krylov sequence $K = K\langle AM^{-1}, r_1\rangle$, and let

$$R = KU, \quad \tilde{R} = K\tilde{U} \qquad \text{where } U, \tilde{U} \in \mathbf{U}^{(1)}$$

be residual sequences. Then $AM^{-1}R = RH$ and $AM^{-1}\tilde{R} = \tilde{R}\tilde{H}$ where by lemma 5 the Hessenberg matrices can be factored

$$H = (J - E_1)V, \quad \tilde{H} = (J - E_1)\tilde{V} \qquad \text{with } V, \tilde{V} \text{ upper triangular.}$$

Use the fact that $RE_1 = \tilde{R}E_1$ to combine this as

$$\begin{array}{l} AM^{-1}KUV^{-1} = R(J-E_1) \\ AM^{-1}K\tilde{U}\tilde{V}^{-1} = \tilde{R}(J-E_1) \end{array} \quad \Rightarrow \quad \tilde{R}J = RJ + AM^{-1}KW,$$

with $W = \tilde{U}\tilde{V}^{-1} - UV^{-1}$. It now follows that

$$\begin{aligned} & J^t\tilde{R}^tA^{-1}\tilde{R}J \\ = \;& J^tR^tA^{-1}RJ + J^tR^tA^{-1}AM^{-1}KW + W^tK^tM^{-t}A^tA^{-1}RJ \\ & + W^tK^tM^{-t}A^tA^{-1}AM^{-1}KW \\ = \;& J^tR^tA^{-1}RJ + J^tR^tM^{-1}KW + W^tK^tM^{-t}RJ + W^tK^tM^{-t}AM^{-1}KW, \end{aligned}$$

where we use the symmetry of A to simplify $A^tA^{-1} = I$. Now, the second and third term are strictly upper and lower triangular because of the upper triangularity of $R^tM^{-1}K = R^tM^{-1}RU^{-1}$. Thus $\tilde{r}_i^tA^{-1}\tilde{r}_i \geq r_i^tA^{-1}r_i$ with equality only if $W = 0$, that is if $\tilde{R} = R$. •

Note that, even though we looked at a preconditioned system, the M-orthogonality caused the minimisation to be in the norm of the original matrix.

6.3 Minimization under general inner product

We will now investigate the minimisation properties of methods based on Arnoldi orthogonalisation. Recall that the polynomial iterative method is completely characterised by

$$\begin{cases} APD = R(I-J), \\ M^{-1}R = P(I-U), \\ R^tN^{-t}R \text{ nonsingularly lower triangular.} \end{cases}$$

There is an equivalence between orthogonality of R and of P: from the coupled two-term recurrences we find that

$$R^tN^{-1}R = R^tN^{-1}MP(I-U),$$

so $P^tM^tN^{-t}R$ is lower triangular, and from this

$$P^tM^tN^{-t}APD = P^tM^tN^{-t}R(I-J) \quad = (I-U)^{-t}R^tN^{-t}R(I-J),$$

so $P^tM^tN^{-t}AP$ is lower triangular, and

$$p_i^tM^tN^{-t}Ap_id_{ii} = r_i^tN^{-t}r_i.$$

We easily see that, given A and M, P^tZP is lower triangular iff $R^tN^{-t}R$ is lower triangular, with N and Z related by $M^tN^{-1}A = Z$.

We will now consider special cases for the choice of N.

$N = I$ This implies that R^tR is diagonal and P^tM^tAP lower triangular.

$N = M^t$ This implies that $R^tM^{-1}R$ is upper triangular and P^tAP lower triangular; in the symmetric case both are diagonal.

$N = A$ This gives $R^tA^{-1}R$ upper triangular and P^tM^tP lower triangular; in the symmetric case both are diagonal.

$N = MA$ This gives $R^tA^{-t}M^{-1}R$ upper triangular and P^tP diagonal.

$N = MA^{-t}$ This gives $R^tAM^{-1}R$ upper triangular, and P^tA^tAP diagonal. As we know from corollary 36, this minimises the norms of the residuals.

6.4 Lanczos' Minimized iterations

For further reading: the material in this section is not referred to elsewhere.

Theorem 42 *Choosing the sequences R and S to be orthogonal minimizes the inner products $s_i^t r_i$ (modulo some normalization of the sequences).*

Proof: Let X, Y be the Krylov sequences following from $AX = XJ$ and $A^tY = YJ$, and assume that $r_1 \parallel x_1$, $s_1 \parallel y_1$, and $AR = RH$, $A^tS = SH$ for some upper Hessenberg matrix H, such that R^tS is diagonal with positive diagonal elements.

Let further $\tilde{r}_1 = r_1$, $\tilde{s}_1 = s_1$ and $A\tilde{R} = \tilde{R}\tilde{H}$, $A^t\tilde{S} = \tilde{S}\tilde{H}$ for some upper Hessenberg matrix $\tilde{H}$. From lemma 9 we find that there are upper triangular matrices U, $\tilde{U}$ such that $R = XU$, $S = YU$, $\tilde{R} = X\tilde{U}$, $\tilde{S} = Y\tilde{U}$. This gives

$$AX = RHU^{-1} = \tilde{R}\tilde{H}U^{-1}; \qquad A^tY = SHU^{-1} = \tilde{S}\tilde{H}U^{-1}.$$

Now if

$$H = L + U_2, \qquad \tilde{H} = \tilde{L} + \tilde{U}_2$$

such that

$$HU^{-1} = J + U_3, \qquad \tilde{H}U^{-1} = J + \tilde{U}_3$$

(see lemma **??** for such Hessenberg matrices) we find

$$\tilde{R}J = RJ + RU_3, \qquad \tilde{S}J = SJ + SU_3.$$

We now get

$$J^t\tilde{S}^t\tilde{R}J = J^tS^tRJ + J^tS^tRU_3 + U_3^tS^tRJ + U_3^tS^tRU_3$$

in which the second and third term are strictly upper and lower triangular respectively. (Here we use for instance that $S^tX = S^tRU^{-1}$ is upper triangular; furthermore, $Y^tX = U^{-t}S^tRU^{-1}$.) Therefore, $\tilde{s}_n^t\tilde{r}_n \geq s_n^t r_n$, with equality only in the case that $\hat{V} = 0$, that is, if $\tilde{R} = R$, $\tilde{S} = S$. •

For the symmetric case of $A = A^t$, $S = R$, this says that the orthogonalizing algorithm minimizes the length of the r_n vectors in each iteration. For the general Lanczos method it gives the minimization of the inner product $s_i^t r_i$, but this implies no minimization for either the r_i or the s_i vectors. This minimization property led Lanczos [47, 48] to name this method 'minimized iterations'. Another name is the 'biconjugate gradient method' [32].

6.5 Line searches

For further reading: the material in this section is not referred to elsewhere.

Consider the minimisation problem

$$\min_x f(x) = \frac{1}{2}x^t Ax - b^t x$$

where A is symmetric positive definite. Taking the derivative $f'(x) = 0$ gives the problem $Ax = b$:

$$f(x+h) - f(x) = \frac{1}{2}(h^t Ax + x^t Ah + h^t Ah) - b^t h = h^t(Ax - b).$$

Thus, solving a linear system is equivalent to minimising a quadratic functional. It is infeasible to find a global minimum, but minimisation along a one-dimensional affine space is possible.

Polynomial iterative methods are a special case of the more general scheme

$$x_{i+1} = x_i + \alpha_i p_i.$$

Here, p_i is called the 'search direction', and the problem of finding the optimal α_i, in the sense that it minimises $f(x_{i+1})$ over the affine space $x_i + \alpha p_i$, is called a 'line search' problem.

If we consider the minimisation problem

$$\min_\alpha f(x+\alpha p) = \min_\alpha \frac{1}{2}\alpha^2 p^t Ap + \frac{1}{2}\alpha(p^t Ax + x^t Ap) + \frac{1}{2}x^t Ax - b^t(x+\alpha p),$$

we find for symmetric A by taking the directional derivative along p with respect to α that

$$\alpha p^t Ap + p^t r = 0 \Rightarrow \alpha = -p^t r / p^t Ap, \tag{41}$$

where $r = Ax - b$. Combining this with equation (**??**), $r_i^t r_i = r_i^t p_i$, we recognise in this the coefficient used in the Conjugate Gradients method. This optimal value of the search parameter is also used in the 'method of steepest descent'; see section 12.6.

In this analysis we used the symmetry of A to write $p^t(A^t - b) = p^t r$. In the nonsymmetric case this obviously doesn't hold. The value for α derived above will still be used in methods considered later in this monograph. However, this is not motivated from minimisation along a line search, but from orthogonality. Under the assumption that p_i is a combination of residuals $r_1 \ldots r_i$, and that x_{i+1} is constructed so that r_{i+1} is orthogonal to the the previous residuals, we find

$$\begin{aligned} x_{i+1} = x_i + p_i\alpha_i \quad &\Rightarrow \quad r_{i+1} = r_i + Ap_i\alpha_i \\ &\Rightarrow \quad 0 = p_i^t r_{i+1} = p_i^t r_i + p_i^t Ap_i\alpha_i \end{aligned}$$

This derivation will be given with more detail in a later section.

Another use of line searches and search directions is in 'seed systems', where the collection of p vectors is built up by solving one linear system, after which they are used for other systems with the same [12, 71], or a similar [13], coefficient matrix; see section 16.12.

7 Orthogonalization

Previous section: 6 Next section: 8

Section 6 discusses the relation between orthogonality and minimization, thus providing the theoretical justification for letting iterative methods be based on orthogonalization. Here we look more constructively at orthogonalization, investigating in particular the conditions under which orthogonality can be satisfied.

Methods such as the Conjugate Gradients algorithm fall under the following broad description:

- Given a matrix A, a Hessenberg matrix $H_{[n,n-1]}$, and a sequence R_n that satisfies some orthogonality condition, related by
$$AM^{-1}R_{n-1} = R_n H_{[n,n-1]},$$
- Find a way to extend
$$R_n \rightarrow R_{n+1}, \quad H_{[n,n-1]} \rightarrow H_{[n+1,n]}$$
while preserving the orthogonality condition.

The study of orthogonalization in iterative methods is then the study of the conditions under which such an extension is feasible.

7.1 Types of orthogonality

The simplest choice of orthogonalisation is to let a residual sequence R, generated by $AM^{-1}R = RH$, be orthogonal under some inner product. However, we will also consider the question of how to let R be semi-orthogonal to another, arbitrary, sequence S.

As indicated above, we extend the R series by suitable construction of the Hessenberg matrix H, which is equivalent to constructing the combinations U such that $R = KU$ where $K = K\langle AM^{-1}, r_1\rangle$.

Definition: General semi-orthogonality condition

Definition 10 *The general semi-orthogonality condition for sequences S, R is*

$$S^t NR \text{ is nonsingularly lower triangular,} \tag{42}$$

where N is a general inner product matrix.

In practice, N will be a combination of A and M^{-1}, and in particular either M^{-1} (corresponding to most simple methods such as CG and BiCG) or AM^{-1} (corresponding to residual minimising methods such as GMRES).

For the relation of this condition to minimisation, see section 6.1, and in particular remark 38. From this condition will follow the scalars in the implementation of the iterative methods; see section 8.2 for the basic methods.

In all of the following, we will consider three cases.

Lanczos orthogonalization This is the case where $S \not\equiv R$ and S itself is based on a Krylov sequence; this leads to the BiConjugate Gradient and Quasi Minimum Residual methods.

Arnoldi orthogonalization This is the case where $S \equiv R$; we distinguish two subcases:

Symmetric case where A and M are symmetric, which leads to the Conjugate Gradient and Minimum Residual methods; and

Nonsymmetric case where at least one of A and M is not symmetric, which leads to methods such as Orthodir and GMRES.

Remark 43 *Equation (42) can also give rise to methods that we do not consider further. For instance, choosing $S = N = I$ makes R lower trian-*

gular, and $AM^{-1}R = RH$ becomes a sort of Gaussian elimination method. This method was proposed by Hessenberg [40]

7.2 Theoretical conditions on orthogonality

We will inductively satisfy the orthogonality condition (42) by constructing r_{n+1} to satisfy

$$S_n^t N r_{n+1} = 0, \qquad s_{n+1}^t N r_{n+1} \neq 0 \tag{43}$$

for all n. Writing the $n+1$-st column of the equation $AM^{-1}R = RH$ as

$$r_{n+1} h_{n+1,n} = AM^{-1} r_n - R_n h_n, \tag{44}$$

we find that the conditions in equation (43) are equivalent to

$$S_n^t NAM^{-1} r_n = S_n^t N R_n h_n \qquad \text{and} \qquad AM^{-1} r_n \neq R_n h_n. \tag{45}$$

We arrive at the following algorithm, which is presented more for theoretical than for practical purposes.

Algorithm: Semi-orthogonalise Krylov sequence to arbitrary sequence

Algorithm 1 *Let A, M, N, r_1 and a sequence S be given. We construct a Hessenberg matrix H and a sequence R such that $AM^{-1}R = RH$, and $S^t NR$ is lower triangular.*

- *Solve the $n \times 1$ vector h_n from*

$$S_n^t NAM^{-1} r_n = S_n^t N R_n h_n.$$

- *Then pick any nonzero value for $h_{n+1,n}$, and define r_{n+1} from*

$$r_{n+1} h_{n+1n} = AM^{-1} r_n - R_n h_n.$$

We will now present the conditions for this process to be defined. First we observe that we are only interested in a particular form of the Hessenberg matrix H:

- we need $h_{n+1n} \neq 0$, since otherwise we are in an invariant subspace R_n, which is typically associated with the iteration being finished or breaking down (see section 3.6); and
- we need $h_{1:n,n} \not\equiv 0$, since otherwise the zero column sum condition on H can not be satisfied.

Lemma 44 *Let sequences R_n and S_n be given, such that*

$$AM^{-1}R_{n-1} = R_n H_{[n,n-1]}$$

and $S_n^t NR_n$ is non-singular lower triangular. Then there exist $r_{n+1} \neq 0$ and $h_{*,n}$ with $h_{n+1,n} \neq 0$ and $h_{1:n,n} \not\equiv 0$ such that

$$AM^{-1}R_n = R_{n+1}H_{[n+1,n]}, \quad \text{and} \quad S_n^t N r_{n+1} = 0 \tag{46}$$

if and only if $S_n^t NAM^{-1}r_n \neq 0$ and R_n is not AM^{-1}-invariant.

Proof: Suppose that $S_n^t NR_n$ is nonsingular. From equation (44) we find that there exist r_{n+1} and h_{n+1n} satisfying the requirements in equation (49) above iff

$$\exists_{h_n \equiv h_{[1:n,n]}} : S_n^t N(AM^{-1}r_n - R_n h_n) = 0 \quad \text{and} \quad AM^{-1}r_n \neq R_n h_n.$$

With $S_n^t NAM^{-1}r_n \neq 0$ we then solve h_n from

$$h_n = (S_n^t NR_n)^{-1} S_n^t NAM^{-1}r_n \tag{47}$$

(see equation (45)), which is not zero because of the $S_n^t NAM^{-1}r_n \neq 0$ condition of the statement.

We are left to satisfy the condition $AM^{-1}r_n \neq R_n h_n$. This last condition is equivalent to R_n not being AM^{-1}-invariant: if R_n is not AM^{-1}-invariant, we have that $AM^{-1}r_n \neq R_n h_n$ for all h_n; if R_n is AM^{-1}-invariant, there is a k_n such that $AM^{-1}r_n = R_n k_n$, so we have

$$S_n^t NAM^{-1}r_n = S_n^t NR_n k_n \Rightarrow h_n = k_n \Rightarrow AM^{-1}r_n = R_n h_n.$$

This concludes the 'if' part of the proof.

On the other hand, let r_{n+1} and $h_{*,n+1}$ exist satisfying equation (49). We get

$$\begin{aligned} S_n^t NAM^{-1}r_n &= S_n^t N r_{n+1} h_{n+1n} + S_n^t NR_n h_{*,n} \\ &= 0 + S_n^t NR_n h_{*,n} \end{aligned}$$

so, from the nonsingularity of $S_n NR_n$, we find that $S_n^t NAM^{-1}r_n \neq 0$. Since $h_{n+1n} \neq 0$, we find from equation (44) that $AM^{-1}r_n \neq R_n h_n$. By the above argument, this implies that R_n is not AM^{-1}-invariant. •

Substituting equation (47) into equation (44), we find

$$r_{n+1}h_{n+1n} = AM^{-1}r_n - R_n(S_n^t NR_n)^{-1} S_n^t NAM^{-1}r_n. \tag{48}$$

This states that r_{n+1} is obtained, up to scaling, by projecting $AM^{-1}r_n$ onto R_n, and the result is N-orthogonal to S_n.

From the result $S_n^t N r_{n+1} = 0$ we see that algorithm 1 is almost inductively defined. For it to be fully defined we need one more condition, namely that the r_{n+1} vector generated not satisfy $s_{n+1}^t N r_{n+1} = 0$. This is called a breakdown condition for the algorithm, and we will investigate it further in section 7.3.

We tie all of the above up in a theorem:

Theorem 45 *Let S_n be any sequence, and R_n be a residual sequence, such that*

$$\begin{cases} AM^{-1}R_{n-1} = R_n H_{[n,n-1]}, \quad H_{[n,n-1]} \text{ irreducible} \\ \text{and } S_n^t NR_n \text{ is non-singular lower triangular,} \end{cases}$$

and $S_n^t NAM^{-1}r_n \neq 0$. We can extend R and H to

$$\begin{cases} AM^{-1}R_n = R_{n+1}H_{[n+1,n]}, \quad H_{[n+1,n]} \text{ irreducible} \\ S_{n+1}^t NR_{n+1} \text{ is non-singular lower triangular} \end{cases}$$

with R_{n+1} a residual sequence iff R_n is not AM^{-1}-invariant, and the vector r_{n+1} defined by equation (48) does not make the condition $s_{n+1}^t N r_{n+1} = 0$ occur.

Proof: Let $S_n^t N R_n$ be non-singular lower triangular, then by the above lemma, and the condition $S_n^t N A M^{-1} r_n \neq 0$, we can extend R and H iff R_n is not AM^{-1}-invariant. From equation (48), r_{n+1} is determined up to its norm; from lemma 44 we know that $H_{[n+1,n]}$ is such that it can have zero column sums by proper choice of h_{n+1n}. Then R_{n+1} is a residual sequence.

We have $S_n^t N r_{n+1} = 0$, so if $s_{n+1}^t N r_{n+1} \neq 0$, $S_{n+1}^t N R_{n+1}$ is again nonsingular lower triangular. •

The condition that r_{i+1} is orthogonal to the i-dimensional space spanned by S_i is often called a 'Galerkin condition', or a 'Petrov-Galerkin condition'. The connection between Petrov-Galerkin conditions and minimisation is discussed in section 6.

7.3 Breakdown conditions

From theorem 45 we see that there are three ways in which the process of generating an orthogonal residual sequence can halt.

- R_n is AM^{-1}-invariant.
- $S_n^t N A M^{-1} r_n = 0$.
- $S_n^t N r_n = 0$.

We know from corollary 22 that the first condition coincides with finding the solution to the linear system. It remains to consider the second and third condition in more detail. As an aside we note that, traditionally, iterative methods were derived in unpreconditioned form, that is, $N = M = I$. In that case, the third condition ceases to be a problem, since $r_n = 0$ implies that the solution has been found.

7.3.1 Arnoldi orthogonalization under M^{-1} inner product; symmetric case

The Conjugate Gradient method is a case of Arnoldi orthogonalization (that is, $S \equiv R$) in the symmetric case, and using the M^{-1} inner product. Assuming nonsingularity – and in particular under the assumption of positive definiteness – we have that $x^t A x \neq 0$ and $x^t M^{-1} x \neq 0$ for all nonzero x.

Clearly, from symmetry, both $S_n^t N r_n = R_n^t M^{-1} r_n$ and $S_n^t N A M^{-1} r_n = R_n^t M^{-1} A M^{-1} r_n$ are nonzero, so the algorithm will not break down.

7.3.2 Arnoldi orthogonalization under M^{-1} inner product; nonsymmetric case

Methods such as OrthoDir use Arnoldi orthogonalization in the nonsymmetric case, with the M^{-1} inner product. Both the expression $S_n^t N r_n = R_n^t M^{-1} r_n$ and $S_n^t N A M^{-1} r_n = R_n^t M^{-1} A M^{-1} r_n$ can be zero; in the special

case of M pd – in particular in the unpreconditioned case of $M = I$ – only the second expression can be zero. Thus this method can break down.

7.3.3 Arnoldi orthogonalization under AM^{-1} inner product

The MinRes method for symmetric systems, and the GCR, OrthoMin, and GMRES methods for nonsymmetric systems, use Arnoldi orthogonalization, and are equivalent to using the $N^t = AM^{-1}$ inner product (Combine equation (42) and remark 38). Therefore, under the assumption of non-singular A, they do not suffer from the second breakdown condition, and, if M is pd – in particular in the unpreconditioned case of $M = I$ – neither from the third.

However, in practice, breakdown is also dependent on the implementation of the algorithm. As we show in section 9.6, algorithms such as GCR, while mathematically equivalent to GMRES, suffer from breakdown by the second condition, while GMRES does not.

7.3.4 Lanczos orthogonalization

Think hard about this one.

7.4 General Arnoldi orthogonalization

The Arnoldi method corresponds to the case $S \equiv R$ and $N = M^{-1}$, so that $R^t M^{-1} R$ is diagonal. We prove the existence of the sequence R.

Existence of general Arnoldi orthogonalization

Lemma 46 *Let a sequence R_n be given, such that*

$$AM^{-1}R_{n-1} = R_n H_{[n,n-1]}$$

and $R_n^t M^{-1} R_n$ is non-singular lower triangular. Then there exist $r_{n+1} \neq 0$ and $h_{*,n}$ with $h_{n+1,n} \neq 0$ and $h_{1:n,n} \not\equiv 0$ such that

$$AM^{-1}R_n = R_{n+1}H_{[n+1,n]}, \quad \text{and} \quad R_n^t M^{-1} r_{n+1} = 0 \tag{49}$$

if and only if $R_n^t M^{-1} A M^{-1} r_n \neq 0$ and R_n is not AM^{-1}-invariant.

Proof: This is a special case of lemma 44 for $S \equiv R$ and $N = M^{-1}$. •

7.5 Arnoldi orthogonalisation and the Conjugate Gradients method

Question: *This section should use M.*

The classical conjugate gradient method stems from the choice $S = R$ in an Arnoldi method, applied to a symmetric matrix A.

Lemma 47 *The Arnoldi method corresponds to a transformation of A to upper Hessenberg form; if A is symmetric this is a tridiagonalization; if A is positive definite the breakdown condition will not occur.*

Proof: If $S = R$, the lower triangular matrix S^tR is diagonal, and $S^tAR = R^tAR = R^tRH$ is a transformation of A to Hessenberg form. If A is symmetric, $R^tAR = R^tRH = H^tR^tR$, so H is of both upper and lower Hessenberg form, hence tridiagonal. The absence of breakdown follows from the discussion in section 7.3.1. •

Lemma 48 *In the conjugate gradient method for a symmetric matrix A, the Hessenberg matrix H is symmetric iff R is equilibrated, that is, there is an α such that $\forall_i$: $\|r_i\| = \alpha$.*

Proof: Let R be equilibrated with $\|r_i\| \equiv \alpha$, then $R^tAR = R^tRH = \alpha^2 H$, so H is symmetric. For the converse, assume that H is symmetric, and let $D = R^tR$. Since R^tAR is symmetric, we have $DH = H^tD$, so $DHD^{-1} = H^t = H$. From the symmetry of H, $h_{ii+1} = h_{i+1i}$, so $d_{i+1}/d_i = d_i/d_{i+1}$, that is, $\|r_i\| = \|r_{i+1}\|$. •

We draw the trivial conclusion.

Corollary 49 *If the sequence R is normalized in CG for a symmetric matrix, H is symmetric.*

The symmetry of H is used for instance in SVD calculations; see section 16.10.1.

The question whether for unsymmetric matrices A the upper Hessenberg matrix can take on a banded form with a small bandwidth is of practical importance, since such a form limits the length of the recurrences such as (37) or (38). This question was answered largely negatively by [81] and more generally by [31]: the conjugate gradient algorithm gives a tridiagonal matrix if the spectrum of A lies on a line in the complex plane; for other matrices the bandwidth is a large fraction of the matrix size.

7.6 Lanczos orthogonalisation and the BiConjugate Gradients method

So far we have let the sequence S to which R was orthogonalised be arbitrary, except when we took $S = R$. It is easy to show that letting S be based on a Krylov sequence leads to full (bi-)orthogonalisation of R and S. This is the basis of the Lanczos and BiConjugate Gradients methods.

We will limit the inner product to the choice $N = M^{-1}$.

First we show that the semi-orthogonalisation condition that $S^t M^{-1} R$ be lower triangular becomes a bi-orthogonalisation condition where $S^t M^{-1} R$ is diagonal if S is itself based on a Krylov sequence.

Lemma 50 *Let the sequences R, S satisfy*

$$AM^{-1}R = RH, \quad A^t M^{-t} S = S\tilde{H}$$

with H, $\tilde{H}$ upper Hessenberg and assume that $S^t M^{-1} R$ lower triangular. Then $S^t M^{-1} R$ is diagonal, and H, $\tilde{H}$ are tridiagonal.

Proof: We have

$$\tilde{H}^t S^t M^{-1} R = S^t M^{-1} A M^{-1} R = S^t M^{-1} R H.$$

The first term is lower Hessenberg, so $S^t R H$, which is a priori upper Hessenberg or full, is also lower Hessenberg. Therefore both H and $\tilde{H}$ must be tridiagonal, and $S^t M^{-1} R$ diagonal. Additionally, $S^t M^{-1} A M^{-1} R$ is tridiagonal. •

We note that, unlike in the CG case, these conclusions do not depend on symmetry of A and M, or equality of the sequences R and S.

We now have the ingredients for the Bi-conjugate Gradient method. We first give the theoretical algorithm analogous to algorithm 1, where we also solve the columns of K.

Algorithm: Bi-orthogonalisation

Algorithm 2 *Let A, r_1, s_1 be given. We inductively construct Hessenberg matrices H and K and sequences R and S such that $AM^{-1}R = RH$, $A^t M^{-t} S = SK$, and $S^t M^{-1} R$ is nonsingular diagonal.*

- *Solve the $n \times 1$ vector h_n from*

$$S_n^t M^{-1} A M^{-1} r_n = S_n^t M^{-1} R_n h_n.$$

- *Then pick any nonzero value for $h_{n+1,n}$, and define r_{n+1} from*

$$r_{n+1} h_{n+1,n} = AM^{-1} r_n - R_n h_n.$$

- *Solve k_n from*

$$s_n^t M^{-1} A M^{-1} R_n = k_n^t S_n^t M^{-1} R_n.$$

- *Pick any nonzero value for k_{n+1n}, and define s_{n+1} from*

$$k_{n+1,n} s_{n+1} = A^t M^{-t} s_n - S_n k_n.$$

This algorithm is not intended as a practical way of computing the bi-orthogonal sequences S and R. We merely using it to establish the theoretical conditions for the existence of the sequences.

BiCG orthogonalisation

Theorem 51 *Let S_n be a sequence satisfying $A^t M^{-t} S_{n-1} = S_n \tilde{H}_{[n,n-1]}$ with $\tilde{H}$ irreducible upper Hessenberg, and let R_n be a residual sequence satisfying $AM^{-1}R_{n-1} = R_n H_{[n,n-1]}$. We can extend R and H to*

$$\begin{cases} AM^{-1}R_n = R_{n+1} H_{[n+1,n]} \\ S_{n+1}^t M^{-1} R_{n+1} \text{ is non-singular lower triangular} \end{cases}$$

with R_{n+1} a residual sequence iff R_n is not AM^{-1}-invariant, and the condition $s_{n+1}^t M^{-1} r_{n+1} = 0$ does not occur.

Proof: Use corollary 45; by lemma 50 the condition $S_n^t M^{-1} A M^{-1} r_n = 0$ can not occur, since that would imply that $(S_n^t M^{-1} R_n) h_{1:n,n} = 0$, which by the zero column sum requirement of H (theorem 30) would imply $h_{n+1n} = 0$, in other words that H has a zero column. •

Obviously, if $r_1 = s_1$ and A and M are symmetric, then $S_n = R_n$ for all n, and $S_{n+1}^t M^{-1} R_{n+1}$ is singular iff $r_{n+1} = 0$. The case where A is not symmetric, hence $S \not\equiv R$, corresponds to the Lanczos method. We can choose r_1 and s_1 independently, but since there is no practical algorithm for the choice of s_1, usually they are taken the same.

However, there are limitations on the choice of K. In fact, K is completely determined by H and $\Omega = S^t M^{-1} R$, according to the following lemma.

Lemma 52 *The Lanczos method generates Hessenberg matrices H and K that are of tridiagonal form; with $\Omega = S^t M^{-1} R$ they satisfy*

$$\Omega H = K^t \Omega. \tag{50}$$

Proof: We have that $S^t M^{-1} R$ is a diagonal matrix by lemma 50, so for $S^t M^{-1} A M^{-1} R$ we find

$$S^t M^{-1} A M^{-1} R = S^t M^{-1} R H = K^t S^t M^{-1} R.$$

This establishes equation (50). •

7.7 Projection

An orthogonal projection operator, projecting along a subspace V has the properties that

$$PV = 0, \qquad x \perp V \Rightarrow Px = x$$

from which such properties as $P^2 = P$ easily follow. The explicit form of P is

$$P_V : x \to x - V(V^t V)^{-1} V^t x. \tag{51}$$

Just for reference, here is the basic theorem of projections.

Theorem 53 *The projected vector $P_V x$ has minimum norm in the affine space $x + V$.*

Proof: Write $P_V x = x - V\bar{y}$, where $\bar{y} = (V^t V)^{-1} V^t x$, and use the property that $x - V\bar{y} \perp V$. Now, for any y:

$$\begin{aligned} \|x - V(y - \bar{y})\| &= (x - V\bar{y} + Vy)^t (x - V\bar{y} + Vy) \\ &= \|x - V\bar{y}\|^2 + \|Vy\|^2 + 2(Vy)^t (x - V\bar{y}) \\ &\geq \|x - V\bar{y}\|^2 \end{aligned}$$

•

Note the similarities between the proof of this theorem and that of theorem 34.

We can define a general projection along V, and orthogonal to W under an inner product B as

$$P_{V,W,B} : x \to x - V(W^t B V)^{-1} W^t B x, \tag{52}$$

which satisfies

$$P_{V,W,B} V = 0, \qquad x \perp_B W \Rightarrow P_{V,W,B} x = x.$$

Trying to reproduce theorem 53 for this inner product gives us that $\|x - V\bar{y}\|_B$, where $\bar{y} = (W^tBV)^{-1}W^tBx$, is minimal if

$$(Vy)^tB(x - V\bar{y}) + (Vy)^tB^t(x - V\bar{y}) \geq 0$$

which is true (with equality to zero) if B is symmetric, so that both terms are equal, and $V = W$, so that we can use the B-orthogonality of $x - V\bar{y}$ to W.

We repeat the observation of equation (48) that

$$r_{n+1}h_{n+1n} = AM^{-1}r_n - R_n(S_n^tN^{-t}R_n)^{-1}S_n^tNAM^{-1}r_n,$$

which states that r_{n+1} is obtained by projecting $AM^{-1}r_n$ along R_n, and the result is N-orthogonal to S_n.

Question: *Generalise this for other inner products.*

8 The iterative formulation of the conjugate gradient and Lanczos methods

Previous section: 7 Next section: 9

In chapter 7 we derived in the abstract conditions on the existence of various iterative schemes. We will now derive in detail the scalar coefficients in the actual recurrences, in particular we will derive the quantities d_{ii} and u_{ii+k} in equation (38). The derivations will follow from the orthogonality conditions of the methods, that is, for Arnoldi orthogonalisation

$$r_i^tM^{-1}r_j = 0 \qquad \text{for } i \neq j, \tag{53}$$

and for Lanczos orthogonalisation

$$s_i^tM^{-1}r_j = 0 \qquad \text{for } i \neq j. \tag{54}$$

The coefficients h_{ij} for the three-term recurrence formulation in equation (37) will be derived in section 8.3.

8.1 Search directions and the solution of linear systems

In item 5 of theorem 32 we introduced the idea of splitting the generating recurrence for the residuals R in coupled recurrences for R and the search directions P. For the individual elements of the R and P sequences we find the recurrences:

$$r_{i+1} = r_i - Ap_id_i, \qquad p_i = M^{-1}r_i + \sum_{j<i} p_ju_{ji},$$

where the constants d_i and u_{ji} follow from orthogonality relations for the p_i and r_i vectors, all derived from the orthogonality conditions equation (53) and equation (54).

In the coupled recurrrences for the R and P sequences we recognize the classical formulation of the conjugate gradient method in the case that the generating equation for p_i is a two-term recurrence.

Since the residuals r_i are orthogonal, for some value n we will have $r_n = 0$. For that n, x_n is the solution of the linear system $Ax = f$. Note that this is the theoretical exact convergence of the conjugate gradient method, discovered by [41]; in practice the method usually converges to a given tolerance in far fewer iterations. This 'speed of convergence' depends mostly on the eigenvalues of the matrix; see for instance [5] or [74].

8.2 Derivation of scalars under Arnoldi and Lanczos orthogonalisation

We derive the scalars in the Conjugate and BiConjugate Gradients algorithms, that is, in methods based on orthogonalising R, and bi-orthogonalising R against another sequence S.

In this section we consider the case of coupled residual / search direction sequences, which reduces to coupled two-term recurrences in the cases of symmetric CG and of BiCG.

With the usual factorisation $H = (I-J)D^{-1}(I-U)$ of the Hessenberg matrix, we have the relations for the left and right search directions and residuals:

$$APD_R = R(I-J), \quad M^{-1}R = P(I-U_R);$$
$$A^tQD_R = S(I-J), \quad M^{-t}S = Q(I-U_S),$$

where the second pair is only needed for the BiConjugate Gradient method, and in the scalar case $U_R = U_S$ and $D_R = D_S$.

We distinguish between three cases: symmetric CG, nonsymmetric CG, and BiCG. Here are the structural differences between the methods.

1. BiCG (Lanczos orthogonalization): here we know from lemma 52 that H is tridiagonal, and hence $I-U$ is upper bidiagonal.
2. CG (Arnoldi orthogonalization) in the symmetric case of $A = A^t$ and $M = M^t$: we know from lemma 47 that H is tridiagonal and hence $I-U$ upper bidiagonal. Furthermore, we can immediately derive CG results from the corresponding BiCG ones by substituting $Q \rightarrow P$ and $S \rightarrow R$.
3. Nonsymmetric CG (Arnoldi orthogonalization): here H is of general upper Hessenberg shape, and $I-U$ is a general upper triangular matrix. We can not, in this case, substitute $Q \equiv P$: in effect, we have the relation

$$Q(I-U) = M^{-1}S$$

instead of the above.

All three methods derive their particular scalars from the orthogonality condition that $S^t M^{-1} R$ is diagonal. This is the practical form of the orthogonality equation (42) that we studied in section 7.1.

In preparation for the block case (section 8.4) we will not assume commutativity or even transposability of scalars. Any slight increase in complexity of the exposition is far outweighed by the generality of the results derived.

With the usual factorisation $H = (I-J)D^{-1}(I-U)$ of the Hessenberg matrix, there are two sets of scalars that we need to know: the quantities d_{ii}, and $u_{i-k,i}$ where for symmetric CG (Arnoldi orthogonalization in the symmetric case) and for BiCG (Lanczos orthogonalization) only $k = 1$ is needed.

8.2.1 Coefficients under Arnoldi orthogonalization

We start with a general discussion of orthogonality properties of Arnoldi methods, and the coefficients that follow from them. We will then move on to the special case of a symmetric coefficient matrix.

The Arnoldi method follows from the basic Krylov equations

$$APD = R(I-J), \qquad M^{-1}R = P(I-U)$$

by positing that $\Omega \equiv R^t M^{-1} R$ is diagonal[7].

Question: *Can we do a story like this for $N^{-1} = AM^{-1}$?*

7. It is enough for the following results to require $R^t M^{-1} R$ to be upper triangular. I have not found results that distinguish between the two. The upper triangularity condition arises from minimisation in section 6.1.

First we derive that

$$\begin{array}{l} R^t P(I-U) = R^t M^{-1} R = \Omega \text{ is diagonal} \\ \Rightarrow P^t R \text{ is lower triangular} \\ \text{also: } \mathrm{diag}(P^t R) = \Omega \end{array} \tag{55}$$

Remark 54 *The lower triangularity of $P^t R$ implies that p_i is orthogonal to r_j for $j < i$, or A-orthogonal to e_j. In other words, the p_i component of the solution is exact.*

Remark 55 *The fact that $R^t P$ is upper triangular immediately leads to the computability of the columns of U: they can be solved individually from the relation $R^t P(I-U) = \Omega$. However, in order to derive the classical formulas for U, we have to continue our analysis a bit further.*

Next we find[8] that

$$\begin{array}{l} P^t AP = P^t R(I-J)D^{-1} \\ \Rightarrow P^t AP \text{ is lower triangular} \\ \text{so } \mathrm{diag}(P^t AP) = \mathrm{diag}(P^t R)D^{-1} = \Omega D^{-1} \end{array} \tag{56}$$

In the symmetric case this last equation becomes the defining equation for the coefficients of D:

$$P^t AP = \Omega D^{-1} \text{ or } d_{ii} = r_i^t M^{-1} r_i / p_i^t A p_i \tag{57}$$

We now combine these results to find

$$\begin{array}{rcl} M^{-1}R & = & P(I-U) \\ \Omega & = & R^t P(I-U) \\ D^{-t}(I-J^t)\Omega & = & P^t A^t P(I-U) \\ \text{symmetric case:} & = & D^{-t}\Omega(I-U) \end{array}$$

8. We remark parenthetically that our block framework yields the various orthogonality relation without the usual 'lengthy inductive argument[s]' [50].

which in the symmetric case leads to a simple equation for the coefficents of U:

$$\Omega^{-1}(I-J^t)\Omega = (I-U) \text{ or } u_{ii+1} = \omega_{i+1}/\omega_i. \tag{58}$$

Summarising, the Conjugate Gradient algorithm for symmetric matrices is based around the following two recurrences:

$$r_{i+1} = r_i - Ap_i\alpha_i, \quad \alpha_i = (p_i^t Ap_i)^{-1}(r_i^t M^{-t} r_i), \tag{59}$$

which is the symmetric case of equation (64), and

$$p_{i+1} = M^{-1}r_{i+1} + p_i(r_i^t M^{-1} r_i)^{-1}(r_{i+1}^t M^{-1} r_{i+1}). \tag{60}$$

which uses equation (66).

Algorithm: Conjugate Gradients Method

Algorithm 3 *In order to solve $Ax = f$, choose x_1 arbitrarily. Let $r_1 = Ax_1 - f$. Then perform the following steps for $i = 1, \ldots$:*

- *Perform a stopping test on $\|r_i\|$.*
- *Compute the preconditioned residual*

  $$M^{-1}r_i$$

  and the inner product $r_i^t M^{-1} r_i$.
- *Compute the new search direction as $p_1 = M^{-1}r_1$, and for $i > 1$*

  $$p_i = M^{-1}r_i + p_{i-1}u_{i-1,i},$$

  where $u_{i-1i} = r_i^t M^{-1} r_i / r_{i-1}^t M^{-1} r_{i-1}$.
- *Compute the A product with the search direction*

  $$Ap_i$$

  and the inner product $p_i^t Ap_i$.
- *Now update the iterate*

  $$x_{i+1} = x_i - p_i d_{ii}$$

  and the residual

  $$r_{i+1} = r_i - Ap_i d_{ii}$$

  where $d_{ii} = r_i^t M^{-1} r_i / p_i^t Ap_i$.

8.2.2 Arnoldi orthogonalisation in the nonsymmetric case

Likewise, in the case of nonsymmetric CG, we find from

$$\begin{aligned} P^t A^t M^{-1} R &= P^t A^t P(I-U) \\ &= D^{-t}(I-J)^t R^t M^{-1} R \end{aligned}$$

that

$$D^{-t}(I-J^t)R^t M^{-1} R = P^t A^t P(I-U). \tag{61}$$

In equation (61) we find for the $n+1$-st column of U

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ -d_n^{-t} \\ d_{n+1}^{-t} \end{pmatrix} r_{n+1}^t M^{-1} r_{n+1} = \begin{pmatrix} p_1^t A^t p_1 & \cdots & p_1^t A^t p_{n+1} \\ & \ddots & \vdots \\ & & p_{n+1}^t A^t p_{n+1} \end{pmatrix} \begin{pmatrix} -u_{1n+1} \\ \vdots \\ -u_{nn+1} \\ 1 \end{pmatrix} \tag{62}$$

from which first of all we find by considering the last row[9]

$$\begin{aligned} d_{n+1}^{-t} r_{n+1}^t M^{-1} r_{n+1} &= p_{n+1}^t A^t p_{n+1} && (63)\\ d_{n+1}(r_{n+1}^t M^{-t} r_{n+1})^{-1} &= (p_{n+1}^t A p_{n+1})^{-1} \\ d_{n+1} &= (p_{n+1}^t A p_{n+1})^{-1}(r_{n+1}^t M^{-t} r_{n+1}) && (64)\end{aligned}$$

Eliminating the last row and column in equation (61) we find, using equation (63) above,

$$\begin{pmatrix} & & 0 \\ & & \vdots \\ & & 0 \\ (r_n^t M^{-1} r_n)^{-1} r_{n+1}^t M^{-1} r_{n+1} & & \end{pmatrix} + (p_n^t A p_n)^{-1} \begin{pmatrix} p_1^t A^t p_{n+1} \\ \vdots \\ p_n^t A^t p_{n+1} \end{pmatrix}$$

$$= (p_n^t A^t p_n)^{-1} \begin{pmatrix} p_1^t A^t p_1 & \dots & p_1^t A^t p_n \\ & \ddots & \vdots \\ & & p_n^t A^t p_n \end{pmatrix} \begin{pmatrix} u_{1n+1} \\ \vdots \\ u_{nn+1} \end{pmatrix} \qquad (65)$$

which reduces immediately to

$$u_{nn+1} = (r_n^t M^{-1} r_n)^{-1}(r_{n+1}^t M^{-1} r_{n+1}), \qquad (66)$$

for the case of a symmetric system[10]. We note that this equation would also be obtained by making the identifications $Q \equiv P$, $S \equiv R$ in equation (75).

The method we have described in this section is sometimes called the Full Orthogonalisation Method (FOM); it reduces to the ordinary CG method in the case of a symmetric matrix. Various methods have been proposed in the literature implementing Arnoldi orthogonalisation under the M^{-1} inner product – in fact, most authors use unpreconditioned methods in their expositions – in the nonsymmetric case: OrthoRes [44] and GENCG [16, 85].

8.2.3 Lanczos case

We start with the basic premisse:

$$S^t M^{-1} R \qquad \text{is diagonal.}$$

Observing that $S^t M^{-1} A M^{-1} R = S^t M^{-1} R H = H^t S^t M^{-1} R$ is of both upper and lower Hessenberg form:

$$S^t M^{-1} A M^{-1} R \qquad \text{is tridiagonal.}$$

From $S^t P(I-U) = S^t M^{-1} R$:

$$S^t P \qquad \text{is upper triangular,}$$

and likewise from $(I-U)^t Q^t R = S^t M^{-1} R$

$$Q^t R \qquad \text{lower triangular,}$$

and from the last two, noting that by $Q^t A P = Q^t R(I-J)D^{-1}$ we find $Q^t A P$ to be lower triangular, and by $Q^t A P = D^{-t}(I-J^t) S^t P$ upper triangular,

$$Q^t A P \qquad \text{diagonal.}$$

---

9. The value for d_{ii} can also be derived directly from
$$\begin{aligned} \mathrm{diag}(P^t A P D) &= \mathrm{diag}(P^t R(I-J)) = \mathrm{diag}(P^t R) = \mathrm{diag}((I-U)^{-t} R^t M^{-t} R) \\ &= \mathrm{diag}(R^t M^{-t} R) \end{aligned}$$
hence
$$\begin{aligned} p_i^t A p_i d_i &= p_i^t r_i = r_i^t M^{-t} r_i \\ \Rightarrow d_i &= (p_i^t A p_i)^{-1}(r_i^t M^{-t} r_i) \end{aligned}$$

10. In the symmetric case we can derive u_{ii+1} from investigating the relation $R^t M^{-t} A M^{-1} R = R^t M^{-1} R H$. In the lower triangle we find
$$\begin{aligned} (i+1,i): \qquad & r_{i+1}^t M^{-t} A M^{-1} r_i = -r_{i+1}^t M^{-1} r_{i+1} d_i^{-1} \\ \Rightarrow \quad & r_i^t M^{-t} A^t M^{-1} r_{i+1} = -d_i^{-t} r_{i+1}^t M^{-t} r_{i+1}, \end{aligned}$$
which result we use (with the symmetry of A) for the following relation in the upper triangle:
$$\begin{aligned} (i,i+1): \qquad & r_i^t M^{-t} A M^{-1} r_{i+1} = (r_i^t M^{-1} r_i) d_{i+1}^{-1} u_{ii+1} \\ \Rightarrow \quad u_{ii+1} &= -d_i (r_i^t M^{-1} r_i)^{-1} d_i^{-t} (r_{i+1}^t M^{-1} r_{i+1}) \\ &= -(p_i^t A p_i)^{-1} d_i^{-t} (r_{i+1}^t M^{-1} r_{i+1}) \\ &= -(p_i^t A p_i)^{-1} (p_i^t A p_i)(r_i^t M^{-1} r_i)^{-1} (r_{i+1}^t M^{-1} r_{i+1}) \\ &= -(r_i^t M^{-1} r_i)^{-1} (r_{i+1}^t M^{-1} r_{i+1}) \end{aligned}$$

8.2.4 Lanczos bi-orthogonalisation

In the case of BiCG, we need to find the coefficients for the left and right sequences.

In algorithm 2 the choice for h_{n+1n} and k_{n+1n} was left unspecified. Under the choice $-\sum_{i\leq n} h_{in}$ and $-\sum_{i\leq n} h_{in}$ respectively, both R and S become residual sequences, and we can derive further relations between H and K. If R is a residual sequence, H allows a factorisation (see lemma 25 and theorem 30)

$$H = (I-J)D_R^{-1}(I-U_R).$$

Since $K^t = \Omega H \Omega^{-1}$, we find

$$K = (I-\Omega^{-t}U_R^t\Omega^t)\Omega^{-t}D_R^{-t}\Omega^t(I-\Omega^{-t}J^t\Omega^t).$$

If K is a residual sequence, it too must allow a factorisation

$$K = (I-J)D_S^{-1}(I-U_S),$$

so

$$\Omega^{-t}U_R^t\Omega^t = J,$$

from which we find

$$(s_{n+1}^t M^{-1} r_{n+1})^{-t} u_{R;nn+1}^t (s_n^t M^{-1} r_n)^t = I,$$

that is,

$$u_{R;nn+1} = (s_n^t M^{-1} r_n)^{-1}(s_{n+1}^t M^{-1} r_{n+1}). \tag{67}$$

Likewise,

$$u_{S;nn+1} = (r_n^t M^{-t} s_n)^{-1}(r_{n+1}^t M^{-t} s_{n+1}). \tag{68}$$

Block statement

The above equations are already in a block form; in the scalar case the coefficients are identical for the left and right sequences.
End of block statement

We find from

$$Q^t A M^{-1} R = \begin{cases} Q^t A P (I-U_R) \\ D_S^{-t}(I-J)^t S^t M^{-1} R \end{cases}$$

that

$$D_S^{-t}(I-J^t)S^t M^{-1} R = Q^t A P (I-U_R). \tag{69}$$

and from

$$P^t A^t M^{-t} S = \begin{cases} P^t A^t Q (I-U_S) \\ D_R^{-t}(I-J)^t R^t M^{-t} S \end{cases}$$

that

$$D_R^{-t}(I-J^t)R^t M^{-t} S = P^t A^t Q (I-U_S). \tag{70}$$

In the scalar case the two equations are the same; in the block case we need both of them.

In equation (69) we find for the $n+1$-st column of U, recalling from section 8.2 that $Q^t AP$ is diagonal,

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ -d_{S;n}^{-t} \\ d_{S;n+1}^{-t} \end{pmatrix} s_{n+1}^t M^{-1} r_{n+1} = \mathrm{diag}(q_i^t A p_i)_{i=1\ldots n+1} \begin{pmatrix} -u_{R;1n+1} \\ \vdots \\ -u_{R;nn+1} \\ 1 \end{pmatrix} \tag{71}$$

and a similar result from equation (70). This first of all gives from row $n+1$

$$d_{S;n+1}^{-t} s_{n+1}^t M^{-1} r_{n+1} = q_{n+1}^t A p_{n+1} \tag{72}$$

$$d_{S;n+1}(r_{n+1} M^{-t} s_{n+1})^{-1} = (p_{n+1}^t A^t q_{n+1})^{-1}$$

$$d_{S;n+1} = (p_{n+1}^t A^t q_{n+1})^{-1}(r_{n+1}^t M^{-t} s_{n+1}). \tag{73}$$

and likewise

$$d_{R;n+1} = (q_{n+1}^t A p_{n+1})^{-1} (s_{n+1}^t M^{-1} r_{n+1}). \tag{74}$$

Furthermore, from row n of equation (71), and using equation (72) above,

$$\begin{aligned} q_n^t A p_n u_{R;nn+1} &= d_{S;n}^{-t} s_{n+1}^t M^{-1} r_{n+1} \\ q_n^t A p_n u_{R;nn+1} &= (q_n^t A p_n)(s_n^t M^{-1} r_n)^{-1} (s_{n+1}^t M^{-1} r_{n+1}) \\ u_{R;nn+1} &= (s_n^t M^{-1} r_n)^{-1} (s_{n+1}^t M^{-1} r_{n+1}). \end{aligned} \tag{75}$$

and similarly

$$u_{S;nn+1} = (r_n M^{-t} s_n)^{-1} (r_{n+1} M^{-t} s_{n+1}). \tag{76}$$

8.2.5 Algorithm for the BiConjugate Gradients method

The BiConjugate Gradients algorithm in the scalar case is based on the recurrences

$$\begin{aligned} r_{n+1} &= r_n - A p_i (p_n^t A^t q_n)^{-1} (r_n^t M^{-t} s_n), \\ s_{n+1} &= s_n - A^t q_i (p_n^t A^t q_n)^{-1} (r_n^t M^{-t} s_n), \end{aligned}$$

and

$$\begin{aligned} p_{n+1} &= M^{-1} r_{n+1} + p_i (s_n^t M^{-1} r_n)^{-1} (s_{n+1}^t M^{-1} r_{n+1}), \\ q_{n+1} &= M^{-t} q_{n+1} + q_i (s_n^t M^{-1} r_n)^{-1} (s_{n+1}^t M^{-1} r_{n+1}). \end{aligned}$$

Algorithm: BiConjugate Gradients Method

Algorithm 4 *In order to solve $Ax = f$, choose x_1 arbitrarily. Let $r_1 = Ax_1 - f$ and choose s_1 arbitrarily, for instance $s_1 = r_1$. Then perform the following steps for $i = 1, \ldots$:*

- *Perform a stopping test on $\|r_i\|$.*
- *Compute the preconditioned residuals*

$$M^{-1} r_i, \qquad M^{-t} s_i$$

and the inner product $s_i^t M^{-1} r_i$.
- *Compute the new search directions as $p_1 = M^{-1} r_1$, $q_1 = M^{-1} s_1$, and for $i > 1$*

$$p_i = M^{-1} r_i + p_{i-1} u_{i-1,i}, \qquad q_i = M^{-t} s_i + q_{i-1} u_{i-1,i}.$$

where $u_{i-1i} = s_i^t M^{-1} r_i / s_{i-1}^t M^{-1} r_{i-1}$.
- *Compute the A products with the search directions*

$$A p_i, \qquad A^t q_i$$

and the inner product $q_i^t A p_i$.
- *Now update the iterate*

$$x_{i+1} = x_i - p_i d_{ii}$$

and the residuals

$$r_{i+1} = r_i - A p_i d_{ii}, \qquad s_{i+1} = s_i - A^t q_i d_{ii}$$

where $d_{ii} = s_i^t M^{-1} r_i / q_i^t A p_i$.

8.2.6 Other definitions of search directions

We could have defined the search directions slightly differently by letting $\tilde{P} = R(I-U)^{-1}$. This gives the coupled equations

$$A M^{-1} \tilde{P} D = R(I-J), \qquad \tilde{P}(I-U) = R.$$

Since the matrices D and U are derived only from H, hence independent of the introduction of $\tilde{P}$, they remain the same as in the ordinary derivation

of Conjugate Gradient methods. Hence, we still have to compute the inner product $p_i^t A p_i = (M^{-1}\tilde{p}_i)^t A (M^{-1}\tilde{p}_i)$. Clearly, we need the vector $M^{-1}p_i$, in addition to the usual $M^{-1}r_i$. We can avoid the cost of an extra preconditioner application by recursively computer $M^{-1}\tilde{p}_i$ from the equation $M^{-1}\tilde{P}(I-U) = M^{-1}R$. We arrive at the following iterative formulation:

$$\begin{aligned} z_i &= M^{-1}r_i \\ p_i &= r_i - p_{i-1}\frac{z_i^t r_i}{z_{i-1}^t r_{i-1}} \qquad \text{omit second term if } i=1 \\ q_i &= z_i - q_{i-1}\frac{z_i^t r_i}{z_{i-1}^t r_{i-1}} \qquad \text{omit second term if } i=1 \\ r_{i+1} &= r_i - Aq_i\frac{z_i^t r_i}{q_i^t A q_i} \end{aligned}$$

The update relation for the search direction can be written as $M^{-1}R = P(I-\Omega^{-1}J\Omega)$, or

$$M^{-1}R\Omega^{-1} = \tilde{P}(I-J), \qquad \tilde{P} = P\Omega^{-1}. \tag{77}$$

Writing the residual update as $R(I-J) = AP(P^tAP)^{-1}\Omega$, this becomes

$$R(I-J) = A\tilde{P}(\tilde{P}^tA\tilde{P})^{-1}.$$

Since the P update in equation (77) is now a BLAS Level 1 AXPY operation, we can use standard libraries and perhaps gain some performance efficiency.

8.2.7 Arnoldi case; AM^{-1} inner product

As we observed in section 6.1, making residuals orthogonal under the AM^{-1} inner product leads to minimisation of their lengths over the subspaces they are part of. We will discuss specific implementations of this orthogonalisation scheme in section 9.

8.2.8 Arnoldi case; General orthogonalisation

So far, we have considered for the orthogonality of R only a few special cases. We will now investigate the general case where

$$R^tN^{-t}R \text{ is upper triangular / diagonal,}$$

with the diagonal case probably only corresponding to a symmetric N. In this orthogonality condition, N can be expressable in terms of A and M, but need not be.

Using again the coupled two-term recurrences

$$APD = R(I-J), \qquad P(I-U) = M^{-1}R,$$

with D and U to be derived from the orthogonality condition, we find that

$$R^tN^{-t}R = R^tN^{-1}MP(I-U),$$

whence $R^tN^{-t}MP$ is upper triangular. With

$$P^tM^tN^{-1}R(I-J) = P^tM^tN^{-t}APD$$

we find that $P^tM^tN^{-1}APD$ is lower triangular, and from considering the diagonal of this and the previous equation we find

$$r_i^tN^{-1}r_i = p_i^tM^tN^{-1}r_i = p_i^tM^tN^{-1}p_id_{ii},$$

so

$$d_{ii}^t = r_i^tN^{-t}r_i/p_i^tA^tN^{-t}Mp_i.$$

As in section 8.2, we derive

$$\begin{aligned} P^tA^tN^{-t}MP(I-U) &= R^tN^{-1}R(I-J)D^{-1} \\ &= P^tA^tN^{-t}R, \\ \Rightarrow P^tA^tN^{-t}MP(I-U) &= D^{-t}(I-J)^tR^tN^{-t}R \end{aligned}$$

and we consider column $n+1$ of this equation, giving which is an equality between upper triangular matrices. We obtain a size $n+1$ system

$$\begin{pmatrix} \begin{cases} p_i^t A^t N^{-t} M p_j & j \geq i, \\ & i,j \leq n+1 \\ 0 & otherwise \end{cases} \end{pmatrix} \begin{pmatrix} -u_{1n+1} \\ \vdots \\ -u_{nn+1} \\ 1 \end{pmatrix} = D^{-t} \begin{pmatrix} (r_1 - r_2)^t N^{-t} r_{n+1} \\ \vdots \\ (r_n - r_{n+1})^t N^{-t} r_{n+1} \\ r_{n+1}^t N^{-t} r_{n+1} \end{pmatrix}.$$

The last line of this confirms our derivation of d_{ii} above. After deleting the last row and column, we are left with the size n system

$$\begin{pmatrix} \begin{cases} p_i^t A^t N^{-t} M p_j & j \geq i, \\ & i,j \leq n \\ 0 & otherwise \end{cases} \end{pmatrix} \begin{pmatrix} u_{1n+1} \\ \vdots \\ u_{nn+1} \end{pmatrix} = \begin{pmatrix} p_1^t A^t N^{-t} r_{n+1} \\ \vdots \\ p_n^t A^t N^{-t} r_{n+1} \end{pmatrix},$$

which can be solved recursively.

8.2.9 Iterating under general inner product

There are several equivalent ways of computing the iteration under general inner product.

Residuals:

$$\begin{aligned} r_{i+1} &= r_i + A p_i d_{ii} \\ N^{-1} r_{i+1} &= N^{-1} r_i + N^{-t} A p_i d_{ii} \\ d_{ii} &= r_i^t N^{-t} r_i / p_i^t A^t N^{-t} M p_i \\ N^{-t} r_{i+1} &= N^{-t} r_i + N^{-t} A p_i d_{ii} \end{aligned}$$

in which the quantity $N^{-t} A p_i$ is not computed anywhere else.

Search directions:

$$p_{i+1} = M^{-1} r_{i+1} + \sum_{j \leq i} p_j u_{ji}$$

$$\begin{aligned} M p_{i+1} &= r_{i+1} + \sum_{j \leq i} M p_j u_{ji} \\ N^{-t} M p_{i+1} &= N^{-t} r_{i+1} + \sum_{j \leq i} N^{-t} M p_j u_{ji} \\ A^t N^{-t} M p_{i+1} &= A^t N^{-t} r_{i+1} + \sum_{j \leq i} A^t N^{-t} M p_j u_{ji} \end{aligned}$$

8.3 Three-term recurrences

Some authors [28, 39], have considered a three-term recurrence for the iterands in the conjugate gradient algorithm, corresponding to the three-term recurrence for the residuals. Such a three-term recurrence is the usual mode of presentation for the Lanczos method and the Chebyshev semi-iterative method [39].

From the splitting $H = (I-J)D^{-1}(I-U)$ of the Hessenberg matrix we find

$$H = \begin{pmatrix} & -d_{n-1}^{-1}u_{n-1n} & \\ -d_{n-1}^{-1} & d_n^{-1} + d_{n-1}^{-1}u_{n-1n} & -d_n^{-1}u_{nn+1} \\ & -d_n^{-1} & \end{pmatrix}.$$

This gives a residual recurrence from $AM^{-1}R = RH$:

$$-r_{n+1}d_{nn}^{-1} + r_n(d_{nn}^{-1} + d_{n-1n-1}^{-1}u_{n-1n}) - r_{n-1}d_{n-1n-1}^{-1}u_{n-1n} = AM^{-1}r_n. \tag{78}$$

with a corresponding update equation for the iterates

$$x_{n+1} = (1 + u_{n-1n}d_{nn}/d_{n-1n-1})x_n - M^{-1}r_n - u_{n-1n}(d_{nn}/d_{n-1n-1})x_{n-1}.$$

Question: *How do you derive the d and u coefficients from just the residuals?*

A three-term recurrence can also directly be derived from the elements of the Hessenberg matrix:

$$r_{i+1}h_{i+1i} + r_ih_{ii} + r_{i-1}h_{i-1i} = AM^{-1}r_i \tag{79}$$

where from orthogonalization properties we get the expressions

$$\begin{aligned} h_{ii} &= \tfrac{r_i^tM^{-1}AM^{-1}r_i}{r_i^tr_i}, \\ h_{ii+1} &= \tfrac{r_i^tM^{-1}AM^{-1}r_{i+1}}{r_i^tr_i}, \quad h_{i+1i} = \tfrac{r_{i+1}^tM^{-1}AM^{-1}r_i}{r_{i+1}^tr_{i+1}}. \end{aligned} \tag{80}$$

Note that the h_{i+1i} are arbitrary normalization factors.

If A is symmetric, the scalar h_{ii+1} can be derived as

$$h_{ii+1} = \frac{r_{i+1}^tr_{i+1}}{r_i^tr_i}h_{i+1i}, \tag{81}$$

which can be shown [60, 61] to be a more stable variant of the method.

Factoring the matrix $H = (I-J)D^{-1}(I-U)$ gives a recurrence

$$\begin{aligned} d_{n+1}^{-1} &= h_{n+1n+1} - h_{n+1n}d_nh_{nn+1} \\ &= h_{n+1n+1} + \begin{cases} h_{nn+1} \\ h_{n+1n}u_{nn+1} \end{cases} \end{aligned}$$

The three-term form (79) is often written as two coupled two-term recurrences

$$r_{i+1}h_{i+1i} + r_ih_{ii} = t_i \qquad \text{where} \qquad t_i = Ar_i - r_{i-1}h_{i-1i}.$$

Although this splitting is of less mathematical significance than the introduction of search directions in theorem 32, item 5, it does allow for a slight variant of the computation by noting that $r_i^tAr_i = r_i^tt_i$, which is used in the computation of h_{ii}. This variant is as stable as the original method [60].

8.4 Block Conjugate Gradients

If more than one linear system is to be solved, and the systems are independent, it is possible to iterate on them simultaneously. For n_b systems to be solved, each vector then becomes an $N \times n_b$ matrix, and each scalar (such as $p_i^t A p_i$) becomes an $n_b \times n_b$ matrix, with division turning into matrix inversion. Thus we need the formulas (59) and (60) that were derived assuming non-commutativity of multiplication.

A further refinement of these relations is needed. The usual orthogonality relations only assure that the n_b-wide block vectors are orthogonal as blocks; the vectors inside each r_i block vector are not. Also, the component vectors may be of widely differing norms, making the matrices $r_i^t M r_i$ and $p_i^t A p_i$ badly scaled. To remedy this, we transform the equations by introducing a block diagonal scaling.

8.4.1 Preliminaries

We first introduce the transformation $p_i = \tilde{p}_i \gamma_i$, giving

$$\begin{aligned} p_{i+1} &= M^{-1} r_{i+1} + p_i (r_i^t M^{-1} r_i)^{-1} (r_{i+1}^t M^{-1} r_{i+1}) \\ \tilde{p}_{i+1}\gamma_{i+1} &= M^{-1} r_{i+1} + \tilde{p}_i \gamma_i (r_i^t M^{-1} r_i)^{-1} (r_{i+1}^t M^{-1} r_{i+1}) \end{aligned} \tag{82}$$

and

$$\begin{aligned} r_{i+1} &= r_i - A p_i (p_i^t A p_i)^{-1} (r_i^t M^{-t} r_i) \\ &= r_i - A\tilde{p}_i \gamma_i (\gamma_i^t \tilde{p}_i^t A \tilde{p}_i \gamma_i)^{-1} (r_i^t M^{-t} r_i) \\ &= r_i - A\tilde{p}_i (\tilde{p}_i^t A \tilde{p}_i)^{-1} \gamma_i^{-t} (r_i^t M^{-t} r_i). \end{aligned}$$

This is the block CG algorithm as it was given by O'Leary [58]. We arrive at a specific choice of γ_{i+1} in equation (82) by splitting the equation in two steps, and first computing an auxiliary vector π_{i+1}:

$$\begin{aligned} \pi_{i+1} &= M^{-1} r_{i+1} + \tilde{p}_i \gamma_i (r_i^t M^{-1} r_i)^{-1} (r_{i+1}^t M^{-1} r_{i+1}) \\ \tilde{p}_{i+1} &= \pi_{i+1} \gamma_{i+1}^{-1} \end{aligned}$$

The second step can for instance correspond to a QR decomposition of π_{i+1}. Note that equation (82) only guarantees that the p_i vectors are A-orthogonal as blocks; letting γ_i follow from a QR decomposition makes the component vectors of each p_i block A-orthogonal too.

We go another step further, introducing the transformation $r_i = \tilde{r}_i \gamma_i$, which gives for the search direction update

$$\tilde{p}_{i+1} = M^{-1}\tilde{r}_{i+1} + \tilde{p}_i (\tilde{r}_i^t M^{-1} \tilde{r}_i)^{-1} \eta_i^t (\tilde{r}_{i+1}^t M^{-1} \tilde{r}_{i+1}) \tag{83}$$

where $\eta_i^{-1} = \gamma_i \gamma_{i+1}^{-1}$, and for the residual update

$$\tilde{r}_{i+1}\gamma_{i+1} = \tilde{r}_i \gamma_i - A\tilde{p}_i (\tilde{p}_i^t A \tilde{p}_i)^{-1} (\tilde{r}_i^t M^{-t} \tilde{r}_i)\gamma_i$$

or, splitting the equation by introducing an auxiliary quantity as before:

$$\begin{aligned} \rho_{i+1} &= \tilde{r}_i - A\tilde{p}_i (\tilde{p}_i^t A \tilde{p}_i)^{-1} (\tilde{r}_i^t M^{-t} \tilde{r}_i) \\ \rho_{i+1} &= \tilde{r}_{i+1}\gamma_{i+1}\gamma_i^{-1} \end{aligned} \tag{84}$$

A computationally feasible form of this is

$$\tilde{r}_{i+1} = \rho_{i+1}\eta_i^{-1}, \qquad \gamma_{i+1} = \eta_i \gamma_i; \tag{85}$$

the choices of η_i now determine the values of γ_i; η_i can for instance be chosen to orthogonalise the r_i sequence.

The original, untransformed, residual is no longer computed, but it can be retrieved as

$$r_{i+1} = \rho_{i+1}\gamma_i.$$

The transformed sequences $\tilde{P}$ and $\tilde{R}$ satisfy a Hessenberg relation

$$A\tilde{P}\tilde{D} = \tilde{R}(I - \tilde{J}), \quad \tilde{P}(I - \tilde{U}) = M^{-1}\tilde{R}$$

with

$$\tilde{D} = GDG^{-1}, \quad \tilde{J} = GJG^{-1}, \quad \tilde{U} = GUG^{-1}.$$

The solution update in terms of the transformed vectors becomes

$$x_{i+1} = x_i - \tilde{p}_i(\tilde{p}_i^t A \tilde{p}_i)^{-1}(\tilde{r}_i^t M^{-t}\tilde{r}_i)\gamma_i. \tag{86}$$

The framework for computing the block Conjugate Gradient algorithm now becomes:

Algorithm: Block Conjugate Gradient Method, version 1

Algorithm 5 *In order to solve $Ax = f$ with f an $N \times k$ block vector, choose x_1 as a block vector of maximal rank. Compute $\rho_1 = Ax_1 - f$. Set $\gamma_1 = I_k$. Then perform the following steps for $i = 1, \ldots$:*
Transform the residual:

$$r_i = \rho_i \eta_i^{-1}; \qquad \text{and set } \gamma_{i+1} = \eta_i \gamma_i \tag{87}$$

Apply the preconditioner:

$$z_i = M^{-1} r_i. \tag{88}$$

If $i = 1$, $p_1 = z_1$; if $i > 1$, compute a new search direction:

$$p_i = z_i + p_{i-1}(r_{i-1}^t z_{i_1})^{-1}\eta_i^t (r_i^t z_i) \tag{89}$$

Compute Ap_i, and $\alpha_i = p_i^t A p_i$; with this, update the solution

$$x_{i+1} = x_i - p_i(p_i^t A p_i)^{-1}(r_i^t z_i)\gamma_i. \tag{90}$$

and the residual:

$$\rho_{i+1} = r_i - Ap_i(p_i^t A p_i)^{-1}(r_i^t z_i). \tag{91}$$

8.4.2 Optimised algorithm

In the above algorithm, the choice of η_i is free. For instance, one could let $r_i\eta_i$ be a QR decomposition of ρ_i, so that $r_i^t r_i = I_k$. The choice $r_i^t z_i = I_k$ leads to a slightly different algorithm, in which we compute $\zeta_i = M^{-1}\rho_i$, and set $z_i = \zeta_i \eta_i^{-1}$ after η_i has been computed from the orthogonality condition[11]. These are the ramifications of all $z_i^t r_i$ terms disappearing:

- In the search direction update, the old direction is multiplied by a triangular matrix, rather than a full one, saving $k^2/2$ vector updates.
- The new orthogonalisation of $\zeta_i^t \rho_i$ can be performed in $k^2/2$ vector updates like that of $\rho_i^t \rho_i$. The latter is trivially symmetric; the former is, depending on the symmetry of M.
- The dot product calculation $r_i^t z_i$ disappears, saving $k^2/2$ inner products.

Algorithm: Block Conjugate Gradient Method, version 2

Algorithm 6 *In order to solve $Ax = f$ with f an $N \times k$ block vector, choose x_1 as a block vector of maximal rank. Compute $\rho_1 = Ax_1 - f$. Set $\gamma_1 = I_k$. Then perform the following steps for $i = 1, \ldots$:*
Apply the preconditioner:

$$\zeta_i = M^{-1}\rho_i \tag{92}$$

Transform the residual:

$$r_i = \rho_i \eta_i^{-1}; \quad z_i = \zeta_i \eta_i^{-1}; \qquad \text{and set } \gamma_{i+1} = \eta_i \gamma_i \tag{93}$$

where η_i is chosen such that $r_i^t z_i = I_k$.
If $i = 1$, $p_1 = z_1$; if $i > 1$, compute a new search direction:

$$p_i = z_i + p_{i-1}\eta_i^t \tag{94}$$

11. In practice, z_i will most likely be computed simultaneously with r_i in the joint orthogonalisation of ρ_i and ζ_i, and no separate step is needed

Compute Ap_i, and $\alpha_i = p_i^t Ap_i$; with this, update the solution

$$x_{i+1} = x_i - p_i(p_i^t Ap_i)^{-1}\gamma_i. \tag{95}$$

and the residual:

$$\rho_{i+1} = r_i - Ap_i(p_i^t Ap_i)^{-1}. \tag{96}$$

8.4.3 Block BiConjugate Gradients

We will now derive a block version of the BiConjugate Gradient method. In this, we will introduce transformed sequences

$$p_i = \tilde{p}_i\gamma_i^r, \quad q_i = \tilde{q}_i\gamma_i^\ell, \quad r_i = \tilde{r}_i\gamma_i^r, \quad s_i = \tilde{s}_i\gamma_i^\ell.$$

The derivation proceeds as above, but now using the BiCG scalars of equations (73), (74), (75) and (76):

$$\begin{aligned}
d_{n+1}^\ell &= (p_{n+1}^t A^t q_{n+1})^{-1}(r_{n+1}^t M^{-t} s_{n+1}) \\
d_{n+1}^r &= (q_{n+1}^t Ap_{n+1})^{-1}(s_{n+1}^t M^{-1} r_{n+1}) \\
u_{nn+1}^\ell &= (r_n M^{-t} s_n)^{-1}(r_{n+1} M^{-t} s_{n+1}) \\
u_{nn+1}^r &= (s_n^t M^{-1} r_n)^{-1}(s_{n+1}^t M^{-1} r_{n+1})
\end{aligned}$$

We introduce the following quantities:

$$\eta_i^r = \gamma_{i+1}^r \gamma_i^{r^{-1}}, \quad \eta_i^\ell = \gamma_{i+1}^\ell \gamma_i^{\ell^{-1}}, \qquad c_i = \gamma_i^r \gamma_i^{\ell^{-1}},$$

(in the symmetric case we will have $\eta_i^\ell \equiv \eta_i^r$ and $c_i \equiv 1$), and vectors

$$\rho_{i+1} = r_{i+1}\gamma_{i+1}^r \gamma_i^{r^{-1}}, \qquad \sigma_{i+1} = s_{i+1}\gamma_{i+1}^\ell \gamma_i^{\ell^{-1}}.$$

Now we find for the right residual:

$$\begin{aligned}
r_{i+1} &= r_i - Ap_i(q_i^t Ap_i)^{-1}(s_i^t M^{-1} r_i) \\
\tilde{r}_{i+1}\gamma_{i+1}^r &= \tilde{r}_i\gamma_i^r - A\tilde{p}_i\gamma_i^r \gamma_i^{r^{-1}}(\tilde{q}_i^t A\tilde{p}_i)^{-1}\gamma_i^{\ell^{-t}}\gamma_i^{\ell t}(\tilde{s}_i^t M^{-1}\tilde{r}_i)\gamma_i^r \\
\tilde{r}_{i+1}\gamma_{i+1}^r \gamma_i^{r^{-1}} &= \tilde{r}_i - A\tilde{p}_i(\tilde{q}_i^t A\tilde{p}_i)^{-1}(\tilde{s}_i^t M^{-1}\tilde{r}_i).
\end{aligned}$$

The left residual

$$\begin{aligned}
s_{i+1} &= s_i - A^t q_i(p_i^t A^t q_i)^{-1}(r_i^t M^{-t} s_i) \\
\tilde{s}_{i+1}\gamma_{i+1}^\ell &= \tilde{s}_i\gamma_i^\ell - A^t\tilde{q}_i\gamma_i^\ell \gamma_i^{\ell^{-1}}(\tilde{p}_i^t A^t\tilde{q}_i)^{-1}\gamma_i^{r^{-t}}\gamma_i^{rt}(\tilde{r}_i^t M^{-t}\tilde{s}_i)\gamma_i^\ell \\
\tilde{s}_{i+1}\gamma_{i+1}^\ell \gamma_i^{\ell^{-1}} &= \tilde{s}_i - A^t\tilde{q}_i^{-1}(\tilde{q}_i^t A\tilde{p}_i)^{-1}(\tilde{s}_i^t M^{-1}\tilde{r}_i).
\end{aligned}$$

For the right search direction

$$\begin{aligned}
p_{i+1} &= M^{-1}r_{i+1} + p_i(s_i^t M^{-1} r_i)^{-1}(s_{i+1}^t M^{-1} r_{i+1}) \\
\tilde{p}_{i+1}\gamma_{i+1}^r &= M^{-1}\tilde{r}_{i+1}\gamma_{i+1}^r \\
&\quad + \tilde{p}_i\gamma_i^r \gamma_i^{r^{-1}}(\tilde{s}_i^t M^{-1}\tilde{r}_i)^{-1}\gamma_i^{\ell^{-t}}\gamma_{i+1}^{\ell t}(\tilde{s}_{i+1}^t M^{-1}\tilde{r}_{i+1})\gamma_{i+1}^r \\
\tilde{p}_{i+1} &= M^{-1}\tilde{r}_{i+1} + \tilde{p}_i(\tilde{r}_i^t M^{-t}\tilde{s}_i)^{-1}\eta_i^{\ell t}(\tilde{r}_{i+1}^t M^{-1}\tilde{s}_{i+1})
\end{aligned}$$

For the left search direction:

$$\begin{aligned}
q_{i+1} &= M^{-t}s_{i+1} + q_i(r_i^t M^{-t} s_i)^{-1}(r_{i+1}^t M^{-t} s_{i+1}) \\
\tilde{q}_{i+1}\gamma_{i+1}^\ell &= M^{-t}\tilde{s}_{i+1}\gamma_{i+1}^\ell \\
&\quad + \tilde{q}_i\gamma_i^\ell \gamma_i^{\ell^{-1}}(\tilde{r}_i^t M^{-t}\tilde{s}_i)^{-1}\gamma_i^{r^{-t}}\gamma_{i+1}^{rt}(\tilde{r}_{i+1}^t M^{-t}\tilde{s}_{i+1})\gamma_{i+1}^\ell \\
\tilde{q}_{i+1} &= M^{-t}\tilde{s}_{i+1} + \tilde{q}_i(\tilde{r}_i^t M^{-t}\tilde{s}_i)^{-1}\eta_i^{rt}(\tilde{r}_{i+1}^t M^{-1}\tilde{s}_{i+1})
\end{aligned}$$

As before, the algorithm proceeds by computing ρ_{i+1} and σ_{i+1} directly, deriving η_i^ℓ and η_i^r from the orthogonality condition, and from this

$$\tilde{r}_{i+1} \leftarrow \rho_{i+1}\eta_i^{r^{-1}}, \quad \tilde{s}_{i+1} \leftarrow \sigma_{i+1}\eta_i^{\ell^{-1}}, \quad \gamma_{i+1}^r = \eta_i^r\gamma_i^r, \quad \gamma_{i+1}^\ell = \eta_i^\ell\gamma_i^\ell,$$

after which p_{i+1} and q_{i+1} are computable.

8.5 SYMMLQ

According to the GMRES paper, SYMMLQ is based on an LQ factorisation of the (tridiagonal) Hessenberg matrix.

9 Minimum and quasi-minimum residual methods: MINRES, GMRES, QMR, TFQMR

Previous section: 8 Next section: 10

In this section we study those iterative methods that effect a minimisation of the residual in each iteration over the subspace generated. Such methods fall in two categories, sometimes only differentiated by computation details.

- We recall from section 6.1, and in particular remark 38, that minimisation of residuals is equivalent to making $R^tAM^{-1}R$ diagonal[12]. Methods such as GCR (section 9.6) and GMRESr (section 9.5.4) compute sequences of residuals and search directions to satisfy this condition.
- Other methods compute a basic sequence, in effect the conjugate gradient method or a generalisation of it to nonsymmetric systems, and take combinations of the residuals and iterates of this to obtain the minimum value.

We lay the groundwork for these two approaches in the next two sections. In section 9.1 we derive conditions for search directions in norm-minimising methods. In section 9.2 we study in the abstract the process of taking affine combinations so as to minimise the residual norm of the result.

9.1 Search directions in norm-minimising methods

We start with the generating equations of all iterative methods that use search directions:

$$P(I-U) = M^{-1}R, \qquad APD = R(I-J).$$

Multiplying the first by R^tA we find that

$$R^tAP(I-U) = R^tAM^{-1}R,$$

12. Upper triangularity would suffice, but this is never used.

which is diagonal, hence R^tAP is diagonal, and

$$r_i^t A p_i = r_i^t A M^{-1} r_i.$$

Multiplying the second equation by P^tA^t we get

$$P^tA^tAPD = P^tA^tR(I-J)$$

where the rhs is lower triangular and the lhs symmetric in structure, hence diagonal. In conclusion we find that P^tA^tAP is diagonal and

$$(Ap_i)^t(Ap_i)d_ii = p_i^tA^tr_i.$$

The coefficients in U now have to be chosen to satisfy the orthogonality of AP.

9.2 Affine combinations of residuals

Several methods have been proposed that generate a vector sequence satisfying a minimization property by first generating the sequence R in the usual way[13], and subsequently taking combinations of this sequence. This procedure can be applied both to sequences satisfying Arnoldi and Lanczos orthogonalisation conditions. In the Arnoldi case, the combinations can be taken so as to minimize the L_2 norm of the residual; in the Lanczos case only a quasi-minimization results.

Let R be a residual sequence satisfying $AM^{-1}R = RH$, where H is a Hessenberg matrix with zero column sums, and construct a sequence G by taking affine combinations, specifically,

$$G = RV_1$$

with V_1 upper triangular with column sums equal to 1. In lemma 18 we saw that G is then itself a residual sequence.

Block statement
We recall equation (30) for the block form of affine combinations:

$$\forall_j \colon \sum_i u_{ij} = I_{k\times k}.$$

End of block statement

Since $V_1(J-E_1)$ is a Hessenberg matrix with zero column sums, by lemma 23 it can be written as $V_1(J-E_1) = (I-J)V_2$. Therefore, for some upper triangular V_3:

$$G(J-E_1) = RV_1(J-E_1) = R(I-J)V_2 = -RHV_3,$$

13. Or at least are equivalent to generating such a vector sequence.

where we used the fact that $H = (I-J)V_4$ for some upper triangular V_4.

Thus we arrive at the following formulation for these combining methods:

$$G(J-E_1) = \begin{cases} -RHV \\ -AM^{-1}RV \end{cases} \quad , \qquad g_1 = r_1 \tag{97}$$

where V is an upper triangular matrix.

Using $RHV = AM^{-1}RV$, the solution update corresponding to equation (97) is

$$Y(J-E_1) = -M^{-1}RV. \tag{98}$$

The sequence G is again a residual sequence as already remarked in lemma 18; in particular $AM^{-1}G = G\bar{H}$, where $\bar{H} = V_1^{-1}HV_1$ is a Hessenberg matrix with zero column sums. This follows from $e^t V_1^{-1} H = e^t H = 0^t$.

Block statement
We remark that for the transition $R(I-J)V_2 = -RHV_3$ we need the fact that H has zero column sums in the strong sense of equation (32).
End of block statement

We could have told the above story in reverse: in equation (98) we recognise the defining equation for polynomial method. In other words, both X and Y are methods for the same problem. By lemma 19 we then find that their residual sequences R and G are affine combinations of each other.

Hessenberg matrices from methods for the same problem are similar

Corollary 56 *Let X and Y be polynomial methods for the same problem with $x_1 = y_1$ and let R and G be their residual sequences and H and $\bar{H}$ the respective Hessenberg matrices, then H and $\bar{H}$ are similar.*

Proof: This follows immediately from $\bar{H} = V_1^{-1}HV_1$. •

9.3 General theory of L_2 minimization methods

9.3.1 Derivation of the coefficient minimization problem

From the choice $g_1 = r_1$, that is, $GE_1 = RE_1$, we find in equation (97) that

$$GJ = RE_1 - RHV = R(E_1 - HV),$$

that is, the particular choice of g_{n+1} is induced by a choice for v_n. We now aim at minimising the norm of the g_i vectors, that is, we consider the minimisation problem

$$\min_{v_n} \|g_{n+1}\|. \tag{99}$$

Let $\Omega = \mathrm{diag}(R^*M^{-1}R)^{1/2}$, and define

$$N = R\Omega^{-1}, \qquad \tilde{H} = \Omega H \Omega^{-1},$$

so that

$$AM^{-1}N = N\tilde{H}, \tag{100}$$

while N consists of combinations of the same Krylov sequence as R.

Depending on the nature of the coefficient matrix we have the following cases regarding Ω:

- In the real case with Arnoldi orthogonalisation, $R^tM^{-1}R$ is diagonal, so
$$N^*M^{-1}N = N^tM^{-1}N = I.$$
- In the real case with Lanczos bi-orthogonalisation, $S^tM^{-1}R$ is diagonal, but $R^tM^{-1}R$ is not.
- In the complex symmetric case, $R^tM^{-1}R$ is diagonal, but $R^*M^{-1}R$ is not.
- In the complex nonsymmetric case, $S^tM^{-1}R$ is diagonal, but neither $S^*M^{-1}R$ nor $R^*M^{-1}R$ nor $R^tM^{-1}R$ are.

If R is an orthogonal sequence, that is, if we use Arnoldi orthogonalisation, we have that $R^*M^{-1}R$ is diagonal, so we find $N^*M^{-1}N = I$. If R is not orthogonal, as in the case of Lanczos orthogonalisation, we only have $\mathrm{diag}(N^*M^{-1}N) = I$, but $N^*M^{-1}N \neq I$; we still have

$$\|N_n\|_{M^{-1},2} \leq \sqrt{n} \tag{101}$$

(where the n subscript denotes that we take the block consisting of the first n columns); see section 9.7.

If we assume orthogonal R, we get the following derivation. For $n \geq 1$, $g_{n+1} - g_1$ is the n-th column of $-RHV$; see equation (97). With $(RHV)_n = R_{n+1}H_nv_n$ this gives

$$\begin{aligned}
\|g_{n+1}\|_{M^{-1},L^2(C^N)} &= \|R_{n+1}e_1 - R_{n+1}H_nv_n\|_{M^{-1},L^2(C^N)} && (102)\\
&= \|\Omega_{n+1}e_1 - \Omega_{n+1}H_nv_n\|_{L^2(C^{n+1})} && (103)\\
&= \left\|\|r_1\|e_1 - \tilde{H}_n\Omega_nv_n\right\|_{L^2(C^{n+1})} && (104)
\end{aligned}$$

(In the more general case of non-orthogonal R, the transition from equation (102) to equation (103) is an inequality $\|\cdot\| \leq \sqrt{n+1}\|\cdot\|$; this will be treated in more detail in section 9.7.)

We now find two formulations of the minimisation problem:

- Observing that H_n can be split as $(J-I)U$, we get from equation (103) the minimisation problem
$$\min_{v_n} \|\Omega_{n+1}(e_1 - (J-I)v_n\|_{L^2(C^{n+1})}.$$
This formulation is used in so-called 'residual smoothing' methods; see section 9.9.
- Making a further substitution $\tilde{v}_n = \Omega_n v_n$, (and $\tilde{V} = \Omega V$), we have replaced the minimisation problem (99) by
$$g_{n+1} = g_1 - AM^{-1}N\tilde{v}_n \qquad \text{where } \tilde{v}_n\colon \min_{v_n}\left\|\|r_1\|e_1 - \tilde{H}_nv_n\right\|_{L^2(C^{n+1})}, \tag{105}$$

and we focus on solving this problem instead. The minimisation problem involving H_n is solved by a QR factorisation; this formulation is used in methods such as MinRes, GMRES, and QMR.

Remark 57 *Equation (105) is a minimisation problem that can be solved regardless of conditions on N. It is only when N is orthonormal that minimising $\tilde{v}_n$ is equivalent to minimising g_{n+1}. This is the basis for methods such as MINRES [63], see section 9.4, and GMRES [69], see section 9.5. We consider the general case where N is not orthonormal, which leads to such methods as QMR [35], in section 9.7.*

The solution of equation (105) is explained in section 9.3.2

Remark 58 *By substituting away the Ω scaling, we have removed all reference to the R sequence in this problem: it is now completely expressed in terms of $\tilde{H}$ and the N sequence, which are related by equation (100). These can be generated directly from r_1, without constructing the rest of the R sequence. This is the Arnoldi similarity transformation; see section 11.1*

9.3.2 Solution of the coefficient minimization problem

We now consider the coefficient minimization problem equation (105):

$$g_{n+1} = g_1 - AM^{-1}N\tilde{v}_n \qquad \text{where } \tilde{v}_n\colon \min_{\tilde{v}_n} \left\| \|r_1\| e_1 - \tilde{H}_n \tilde{v}_n \right\|_{L^2(C^{n+1})}.$$

Let $\tilde{H}_n = Q_n U_n$ be a decomposition into an orthonormal matrix and an upper triangular matrix, where Q_n is of size $(n+1)\times n$ and U_n of size $n \times n$. Define

$$\bar{Q}_n = \begin{bmatrix} & * \\ Q_n & \vdots \\ & * \end{bmatrix}, \qquad \bar{U}_n = \begin{bmatrix} & U_n & \\ 0 & \cdots & 0 \end{bmatrix}$$

where the final column of $\bar{Q}_n$ is chosen to make $\bar{Q}_n$ a square orthonormal matrix, then, again, $\tilde{H}_n = \bar{Q}_n \bar{U}_n$. We now find

$$\left\| \|r_1\| e_1 - \tilde{H}_n \tilde{v}_n \right\|_{L^2(C^{n+1})} = \left\| \|r_1\| q^{(n+1)} - \bar{U}_n \tilde{v}_n \right\|_{L^2(C^{n+1})},$$

where $q^{(n+1)^t} = (q_{11}, \ldots, q_{1n+1})$. If we solve

$$\tilde{v}_n = U_n^{-1} q^{(n+1)}_{1:n} \|r_1\|, \tag{106}$$

the first n components of $\bar{U}_n \tilde{v}_n - \|r_1\| q^{(n+1)}$ are zero, so the minimum over all $\tilde{v}_n$ is attained for this value[14][15], and the value of the minimum is $|q_{1n+1}| \|r_1\|$.

Formula (106) can be summarised for all n as $\tilde{V} = U^{-1}\tilde{Q}^t$, where $\tilde{Q}$ is formed from Q as

$$\tilde{Q}^t = \begin{pmatrix} q_{11} & q_{11} & \cdots & q_{11} \\ & q_{12} & \cdots & q_{12} \\ & \emptyset & \ddots & \vdots \\ & & & q_{1n} \end{pmatrix}.$$

9.3.3 Solution vector update for minimization problems

Above, we found that

$$\|g_{n+1}\|_{M^{-1},L^2(C^N)} = |q_{1n+1}| \|r_1\|,$$

14. QR decompositions are unique up to scaling as QD and $D^{-1}R$ where $|d_{ii}| = 1$. In equation (106) this scaling drops out again, so $\tilde{v}_n$ is uniquely determined.
15. This argument is still not airtight.

so for the purposes of a stopping test on this norm we do not explicitly need to form the solution vector or the residual. Instead, these can be formed *in toto* when the decision to terminate the iteration is made.

In case formation of the solution is needed, the following applies.

From $G(J-E_1) = -AM^{-1}N\tilde{V}$ we first of all find a defining formula for the iterates sequence:

$$X(J-E_1) = -M^{-1}N\tilde{V} = -M^{-1}NU^{-1}\tilde{Q}^t. \tag{107}$$

For an updating formula for X we find the even simpler formula

$$X(J-I) = -M^{-1}P\,\mathrm{diag}(q_{1i}), \qquad P = NU^{-1} \tag{108}$$

The elements of P can be easily updated from the r_n vectors, in the case of QMR with a three-term recurrence. In GMRES updating p_n requires all previous such vectors to be stored. Hence, people have considered truncated or restarted versions of the method.

Furthermore, for the minimised residuals G we find similarly:

$$G(J-I) = -ANU^{-1}\mathrm{diag}(q_{1i}) = -N\tilde{H}U^{-1}\mathrm{diag}(q_{1i}) = -NQ\mathrm{diag}(q_{1i}).$$

9.3.4 Relation of residual norm to original iterations

In [17] it is shown that

$$\|r_F^{(k)}\|_2 = \frac{\|r_G^{(k)}\|_2}{\sqrt{1-(\|r_G^{(k)}\|_2/\|r_G^{(k-1)}\|_2)^2}}.$$

9.3.5 Stagnation of the minimised iterations

Investigate the meaning of $\|r_{\mathrm{gmres}}^{(n+1)}\| = \|r_{\mathrm{gmres}}^{(n)}\|$.

9.4 MINRES: The Minimum Residual method

In the above we derived how, given a sequence R satisfying $AM^{-1}R = RH$, one derives affine combinations that mimise some norm. According to section 9.2, taking such combinations and applying them to a residual sequence, gives again a residual sequence. In making the transition from the residual minimisation problem equation (99) to the coefficient minimisation problem equation (105) we used the orthogonality of R.

We know from section 7.5 that, with A and M symmetric, R can be made orthogonal with a three-term recurrence. The resulting minimisation method is known as MINRES [63].

Question: *What is LSQR?*

9.5 GMRES: The Generalised Minimum Residual method

Question: *what is the relation between the gmres hessenberg matrix and the one that governs the residuals?*

Question: *can we do something with scaling or combinations of the arnoldi sequence, instead of the minimisation?*

9.5.1 The Saad and Schultz algorithm

The GMRES, for Generalized Minimal Residual [69], method is found by applying the above minimisation and using the substition of remark 58. Thus the algorithm proceeds as follows:

1. Derive an orthonormal sequence N and upper Hessenberg matrix satisfying
$$AM^{-1}N = NH, \qquad N^tM^{-1}N = I, \qquad n_1 = r_1/\|r_1\|.$$
More about this in section 11.1.
2. Make a QR decomposition: $H_i = Q_iU_i$ where Q_i is of size $i+1 \times i$. Generate an $i+1$-st column q_{i+1} of Q_i, such that the extended Q_i is again orthonormal; then the residual error is $|q_{1i+1}|\|r_1\|$.
3. If this error is deemed small enough, solve
$$v_i = U_i^{-1}(q_{11},\ldots q_{1i})^t\|r_1\|$$
and update residual and solution
$$x_i = x_1 - M^{-1}N_{*,1:i}v_i; \qquad r_i = r_1 - AM^{-1}N_{*,1:i}v_i.$$
4. If the restart parameter is reached, perform the above update regardless the error estimate, let
$$r_1 \leftarrow r_i, \qquad x_1 \leftarrow x_i,$$
and restart the whole algorithm.

A matlab implementation can be found in section A.1.1.

9.5.2 Breakdown of GMRES

In the above algorithm, there are two opportunities for breakdown:

- if step 1 breaks down, that is the sequence N can not be continued since we have found an invariant subspace, or,
- if in step 3 the matrix U_i is singular.

By the analysis in section 11.1, case 1 implies that we can form the exact solution of the linear system. Case 2 implies that H is not an irreducible upper Hessenberg matrix, which also implies (by theorem 45) that we have found an invariant subspace.

We conclude that GMRES does not break down, other than by finding the true solution.

9.5.3 Implementation of the QR decomposition

In the original GMRES paper [69], Saad and Schultz derived equation (106), and the error estimate in item 2 above, from a specific implementation of the QR decomposition by Givens rotations. However, it follows from the exposition in section 9.3.2 that this is merely an implementation detail. A decomposition by Gram-Schmidt orthogonalisation is equally feasible; see section 11.3.1.

9.5.4 Direct computation of the GMRES vectors

The GMRES method used an indirect method to construct the residuals: it did not use either a recurrence in terms of older residuals, or an update formula using search directions. This does not mean it is not possible to do so; see section 9.6

9.5.5 Left-preconditioned GMRES

Based on corollary 33 we can also generate the Arnoldi vectors from

$$v_1 = M^{-1}(Ax_1 - b), \quad n_1 = v_1/\|v_1\|; \qquad M^{-1}AN = VN.$$

9.6 Residual-minimising Arnoldi methods for the non-symmetric case: GCR, Orthodir, GMRESr

There are several algorithms based on Arnoldi orthogonalization for the nonsymmetric case that minimise the residual norms in each iteration. This makes them mathematically equivalent to GMRES (section 9.5) in the sense that they derive the same iterates and residuals. However, unlike GMRES (see section 9.5.2) they are subject to breakdown.

The Generalized Conjugate Residual method (GCR; see next), OrthoDir [44], OrthoMin [80], Axelsson's method [4], were derived in the 1970s and 1980s but that have by now largely slid into obscurity.

The GCR algorithm [26] can be described as in figure 2. Since this algorithm orthogonalises the Ap_i series to itself, we know from section 6.3 that this is equivalent to making the residuals A-orthogonal (AM^{-1}-orthogonal for preconditioned methods), which by corollary 36 minimises the residuals.

$r_1 = p_1 = Ax_1 - b$
for $i = 1, \ldots$
  let $\alpha_i = r_i^t Ap_i/(Ap_i)^t(Ap_i)$.
  update $x_{i+1} = x_i - \alpha_i p_i$
    $r_{i+1} = r_i - \alpha_i Ap_i$
  let $p_{i+1} = r_{i+1} + \sum_{j \leq i} p_j \beta_{ji}$
    where $\{\beta_{ji}\}$ are chosen
    such that $(Ap_{i+1})^t(Ap_j) = 0$ for $j \leq i$.

Figure 2: Generalized Conjugate Residual method

A simple example of the breakdown of this method is given in [69]: with

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

we get $\alpha_1 = 0$, which leads to $r_2 = r_1$. Continuing this, $p_2 = 0$, so computing α_2 gives a divide-by-zero error.

Van der Vorst and Vuik [75] describe a method along the following lines:

- let $p_n = M^{-1} r_n$ and $q_n = AM^{-1} r_n = A p_n$;
- modify $p_n \leftarrow p_n - \sum_{k<n} c_k p_k$, $q_n \leftarrow q_n - \sum_{k<n} c_k q_k$, such that the q_i family is orthogonal;
- normalize by dividing q_n and p_n by $\|q_n\|$;
- update

$$r_{n+1} = r_n - q_n (q_n^t r_n).$$

In this we recognise the basic GCR scheme with preconditioning incorporated[16]. A practically interesting proposition is to let the preconditioner be an embedded iterative method, such as GMRES itself. In van der Vorst and Vuik's terminology that makes it a 'recursive GMRES' method; Saad calls it 'flexible GMRES', but there is a difference between GMRESr and fGMRES.

Van der Vorst and Vuik propose to solve the breakdown problem by incorporating a 'LSQR-switch', that is, in case of breakdown they propose to let $p_n \leftarrow A^t r_n - \sum \ldots$. Van der Vorst and Chan [77] observe that a sufficient condition for avoiding breakdown ($\|c\|_2 = 0$) is that the norm of the residual at the end of an inner iteration is smaller than the right-hand residual: $\|Az^{(m)} - r_i\|_2 < \|r_i\|_2$.

16. It should be noted that van der Vorst and Vuik's derivation of the method is along completely different lines.

9.7 Quasi-minimisation

The simplifying step from the condition (102) to the practically computable form (104) uses the M^{-1}-orthogonality of R. In methods such as BiCG, which are based on Lanczos bi-orthogonalisation, R is not itself orthogonal. However, defining as before $\Omega = \mathrm{diag}(R^* M^{-1} R)^{1/2}$, and $N = R\Omega^{-1}$ (which satisfies $\mathrm{diag}(N^* M^{-1} N) = I$) we have the inequality

$$\|R_{n+1} x\|_{M^{-1},L^2} \leq \|N\|_{M^{-1},L^2} \|x\|_{L^2} \leq \sqrt{n+1}\,\|x\|_{L^2},$$

where we used the following lemma

Lemma 59 *If N of size $m \times n$ satisfies $\mathrm{diag}(N^* M^{-1} N) = I$, then $\|N\|_{M^{-1}} \leq \sqrt{m}$.*

Proof: Recall the definition of the matrix norm:

$$\|N\| = \min_{\|x\|=1} |Nx|,$$

and, using the Cauchy-Schwarz inequality and the elementary fact that $2xy \leq x^2 + y^2$ for all x and y,

$$\begin{aligned} |Nx|^2_{M^{-1}} &= \sum_{i,j} x_i x_j \bar{n}_i^* M^{-1} \bar{n}_j \\ &\leq \sum_i x_i^2 + 2 \sum_{i<j} x_i x_j \\ &\leq \sum_i x_i^2 + \sum_{i<j} (x_i^2 + x_j^2) = n \sum_i x_i^2 = n. \end{aligned}$$

•

This $\sqrt{n}$ term also appears when we consider the complex symmetric conjugate gradient method, where $R^t M^{-1} R$ is diagonal, but the vector space would demand $R^* M^{-1} R$ to be diagonal for true minimisation.

9.8 QMR

The Quasi-Minimum Residual method of Freund and Nachtigal [35] arises from applying quasi-minimisation to the BiCG method.

9.9 Residual smoothing

Several authors [70, 83, 87] have investigated methods that take an iterate sequence $\{x_i\}$ and improve on it by taking affine combinations of the form

$$y_1 = x_1, \qquad y_{i+1} = (1-\eta_i)x_i + \eta_i y_{i+1}.$$

The scalars η_i are determined in such a way that the residuals are minimised in some sense.

Lemma 60 *The residuals $g_i = Ay_i - b$ corresponding to the smoothed sequence are a residual sequence iff $r_i = Ax_i - b$ is a residual sequence.*

Proof: The g_i residuals are affine combinations of the r_i residuals, hence by lemma 18 also a residual sequence. To see this, we find the following relation between the G and R sequences:

$$G \begin{pmatrix} 1 & \eta_1 - 1 & & \\ & 1 & \eta_2 - 1 & \\ & & \ddots & \ddots \end{pmatrix} = R \begin{pmatrix} 1 & & & \\ & \eta_1 & & \\ & & \eta_2 & \\ & & & \ddots \end{pmatrix}$$

or $GN = RD$ where $e^t N = e^t D$, i.e., N and D have the same column sums. It follows that $G = RDN^{-1}$ where DN^{-1} is an upper triangular matrix with column sums $\equiv 1$. Thus, G is a affine combination of residuals as described in section 9.2, and it is itself again a residual sequence. •

Block statement
In the case of block vectors, the scalars η_i become $k \times k$ blocks. If they are of diagonal form, the block method reduces to a number of scalar methods executed at the same time. In the general case, $e^t N = e^t D$ still holds, so DN^{-1} has colum sums $\equiv 1$ in the stronger sense of equation (30).
End of block statement

In practice, residual smoothing is used to minimise the norm of the newly created g_i residuals. We have already seen methods for this purpose, and in fact they can be reproduced by residual smoothing methods.

The following discussion will pertain both to symmetric and nonsymmetric real and complex systems. Let R be the sequence of right residuals of the conjugate gradient method, and S the left residuals of the bi-conjugate gradient method for nonsymmetric system; for symmetric systems we will implicitly assume $S \equiv R$.

In section 9.2 we showed that any affine combination vector g_n of the residuals $r_1, \ldots, r_n$ satisfies the following formula:

$$g_{n+1} - g_1 = -AM^{-1}R_n v_n = -R_{n+1}H_n v_n$$

for some vector v_n, where $H_n = H_{[n+1,n]}$ is the $(n+1) \times n$ Hessenberg matrix describing the iterative method, and $R_n = (r_1, \ldots, r_n)$. Since $g_1 = r_1 = R_n e_1$ (for any n), this leads to the minimisation problem

$$\min \|g_{n+1}\| = \min \|R_{n+1}(e_1 - H_{[n+1,n]}v_n)\|.$$

This is used to generate minimising or quasi-minimising residuals, as we have seen in section 9.3.1

Since H_n can be factored as $H_n = (J-I)_{[n+1,n]}U$ with

$$J - I = \begin{pmatrix} -1 & & & \\ 1 & -1 & & \emptyset \\ & \ddots & \ddots & \\ \emptyset & & & \end{pmatrix}, \qquad U \text{ upper triangular},$$

there is likewise a vector z_n such that

$$g_{n+1} = R_{n+1}(e_1 - (J-I)_{[n+1,n]}z_n). \tag{109}$$

The minimisation problem to be solved is then

$$\min_{z_n} \|g_{n+1}\| = \min_{z_n} \|R_{n+1}v_{n+1}\|. \tag{110}$$

where we define $v_{n+1} = e_1 - (J-I)z_n$.

We now consider the minimisation problem (110) in more detail. In the symmetric case,

$$\|g_{n+1}\|_{M^{-1},2} = \|R_{n+1}v_{n+1}\|_{M^{-1},2} = \|N_{n+1}\Omega^{1/2}v_{n+1}\|_{M^{-1},2} = \|\Omega^{1/2}v_{n+1}\|_2.$$

In the nonsymmetric case, we get from (101):

$$\begin{aligned} \|g_{n+1}\|_{M^{-1},2} &= \|R_{n+1}v_{n+1}\|_{M^{-1},2} = \|N_{n+1}\Omega^{1/2}v_{n+1}\|_{M^{-1},2} \\ &\leq \|N_{n+1}\|_{M^{-1},2}\|\Omega^{1/2}v_{n+1}\|_2 \\ &\leq \sqrt{n+1}\|\Omega^{1/2}v_{n+1}\|_2. \end{aligned} \tag{111}$$

Hence, solving the minimisation problem

$$\min_{z_n} \|\Omega^{1/2}v_{n+1}\|_2 \tag{112}$$

gives the optimal solution in the Krylov basis, exactly in the symmetric case, up to a factor of $\sqrt{n}$ in the nonsymmetric case. The minimisation problem (112) is also essentially the one considered in [33] for the complex symmetric case, although this theoretical justification was not given there.

In the traditional derivation of the QMR method, the minimisation problem (112) was solved by *QR* factorisations, as in [33, 35]. Here we are deriving the residual smoothing method of solving the same minimisation problem, hence deriving the same iterates. We follow the discussion in [87].

Note that the minimisation problem

$$\tau_k^2 = \min_{z_n} \|\Omega^{1/2}(e_1 - (J-I)z_n)\|_2$$

is of the form $\min_x \|Ax - b\|$. In the complex case, z_n can be complex, but A and b are real, so the solution, given by $x = (A^*A)^{-1}A^*b$ is real too.

An explicit form for the minimum, and the value of v for which it is taken, can be given. With $z_n = (\zeta_1, \ldots, \zeta_n)^t$,

$$\tau_k^2 = \min_z \omega_1(1-\zeta_1)^2 + \omega_2(\zeta_1-\zeta_2)^2 + \cdots + \omega_n(\zeta_{n-1}-\zeta_n)^2 + \omega_{n+1}\zeta_n^2.$$

Changing variables $\xi_1 = 1-\zeta_1$, $\xi_k = \zeta_{k+1} - \zeta_k$ for $k = 2,..,n$, $\xi_{n+1} = \zeta_n$, the minimisation problem becomes

$$\tau_n^2 = \min_{\sum \xi_i = 1} \sum \omega_i \xi_i^2$$

for which the unique minimiser is given by

$$\xi_i = \frac{1/\omega_i}{\sum_j \frac{1}{\omega_j}}$$

and the resulting value of the minimum is

$$\tau_n = \sqrt{\frac{1}{\sum_i^n 1/\omega_i}}.$$

Lemma 61 *The minimimum of $f(x) = \sum \omega_i^2 x_i^2$ under the constraints $\omega_i > 0$, $\sum x_i = 1$ is given by $x_i = \omega_i^{-1} / \sum_j \omega_j^{-1}$.*

Proof: The constraint equation translates to $\sum_i \partial x_i / \partial x_j = 0$ for all j. The case $\omega_i x_i \equiv c$ gives $x_i = c/\omega_i$. With

$$1 = \sum x_i = c \sum \omega_i^{-1} \Rightarrow c = 1/\sum_i \omega_i^{-1}$$

this becomes $x_i = \omega_i^{-1} / \sum_i \omega_i^{-1}$. In this value of x,

$$\frac{\partial f}{\partial x_j} = \sum_i \omega_i^2 \frac{\partial x_i^2}{\partial x_j} = \frac{2}{\sum_i \omega_i^{-1}} \sum_i \omega_i \frac{\partial x_i}{\partial x_j} \begin{cases} \leq \frac{2}{\sum_i \omega_i^{-1}} \omega_{max} \sum_i \frac{\partial x_i}{\partial x_j} = 0 \\ \geq \frac{2}{\sum_i \omega_i^{-1}} \omega_{min} \sum_i \frac{\partial x_i}{\partial x_j} = 0 \end{cases} = 0.$$

•

We thus find that L_2 quasi-minimisation of the norm of the coefficient vector z_k gives coefficients τ_k satisfying

$$\frac{1}{\tau_k^2} = \frac{1}{\tau_{k-1}^2} + \frac{1}{\omega_k}, \quad \tau_1^2 = \omega_1, \qquad \text{where } \omega_k = \|s_k^t M^{-1} r_k\|_2. \tag{113}$$

From the discussion in [87] we know that these τ_k coefficients generate the (quasi) minimising residuals:

$$\tau_{k+1}^{-2} g_{k+1} = \tau_k^{-2} g_k + \omega_k^{-1} r_{k+1}.$$

We summarise the resulting algorithms in figure 3.

Lemma 62 *Let G be a residual smoothing sequence, that is, let $G = RB^{-1}$, where R is a residual sequence and B is upper triangular. If every g_i is the minimum norm combination of r_j $(j \leq i)$, then B is bidiagonal.*

Proof: Let $AR = RH$ be the defining relation for the residual sequence R, let Ω be the diagonal matrix with $\omega_i = \|r_i\|$, let $N = R\Omega^{-1}$ be the normalised residuals, and let $\tilde{H} = \Omega H \Omega^{-1}$ be the Hessenberg matrix satisfying $AN = N\tilde{H}$. Let $\tilde{H} = QU$ and $H = \hat{Q}\hat{U}$ be QR decompositions, and define $D_Q = \mathrm{diag}(\omega_1 q_{qi})$.

With the results from section 9.3.1 and equation 34, we find that for a minimised sequence G:

$$\begin{aligned} G(J_I) &= AR\Omega^{-1}U^{-1}D_Q\|r_1\| \\ &= RH\Omega^{-1}U^{-1}D_Q\|r_1\| \\ &= R(J-I)B_1^{-1}D_Q\|r_1\| \end{aligned}$$

We now have that $G(J-I)D_Q^{-1}B_1 = R(J-I)$. Clearly, $D_Q^{-1}B_1$ is upper bidiagonal. Using equation 33:

$$D_Q^{-1}B_1 e = D_Q^{-1}Q'\Omega(J-I)e = \begin{pmatrix} 1 & * & & \\ 1 & * & * & \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \omega_1 \\ 0 \\ \vdots \end{pmatrix} = \omega_1 e$$

it follows that $D_Q^{-1}B_1$ also has constant row sums. By lemma 26 there is then an upper bidiagonal B_3 such that $GB_3(J-I) = R(J-I)$. From this it follows that G is a residual smoothing method. •

Let A be a matrix, and let M be a preconditioner for A, and let b be given;
let x_1 by arbitrary, and $r_1 = Ax_1 - b$;
let $s_1 = r_1$ in the symmetric case, or arbitrary in the nonsymmetric case.
Iterate for $i = 1, \ldots$:

let $\rho_i = s_i^t M^{-1} r_i$ and $\omega_i = r_i^* M^{-1} r_i$
calculate the smoothed solution and residual:
if $i = 1$, $\tau_1 = \rho_1$, $g_1 = r_1$, $y_1 = x_1$
otherwise, $\tau_i = 1/(\tau_{i-1}^{-1} + \omega_i^{-1})$ and

$$y_i = \tau_i(\tau_{i-1}^{-1} y_{i-1} + \omega_i^{-1} x_{i-1})$$
$$g_i = \tau_i(\tau_{i-1}^{-1} g_{i-1} + \omega_i^{-1} r_{i-1})$$

calculate search direction and update the original residual:
if $i = 1$, $p_i = r_i$, ($q_1 = s_1$ in the nonsymmetric case), $\tau_1 = \omega_1$
otherwise, $\beta = \rho_i/\rho_{i-1}$

$$p_i = r_i + \beta p_{i-1}$$
$$q_i = s_i + \beta q_{i-1} \text{ (nonsymmetric only)}$$

with $\alpha = \rho_i / p_i^t A p_i$
update $x_{i+1} = x_i - \alpha p_i$
and $r_{i+1} = r_i - \alpha A p_i$

$$s_{i+1} = s_i - \alpha A^t q_i$$

Figure 3: The residual smoothed conjugate or bi-conjugate gradient algorithm, for real and complex systems.

9.10 TFQMR

Applying residual smoothing to the CGS method (see section 10) gives the Transpose-Free QMR method of Freund [34].

10 Polynomial squaring methods

Previous section: 9 Next section: 11

The Lanczos method can be characterised in general through right and left sequences R and S, generated by

$$AM^{-1}R = RH, \qquad A^tM^{-t}S = SK$$

where H and K are upper Hessenberg matrices. (In the scalar case, H and K are equal; in the block case see equations (67) and (68).) For the vectors in the sequences we have $r_n = \rho_n(AM^{-1})r_1$ and $s_n = \tilde{\rho}_n(A^tM^{-t})s_1$, where ρ_n and $\tilde{\rho}_n$ are $n-1$-st degree polynomials. (Again, in the scalar case for BiCG, $H = K$ and $\rho_n \equiv \tilde{\rho}_n$; see section 7.6 for the exact results that also hold in the block case.) Since presumably both the left and right residuals tend to zero, and

$$s_n^tM^{-1}r_n = (\rho_n(A^tM^{-t})s_1)^tM^{-1}(\rho_n(A)r_1) = s_1^tM^{-1}(\rho_n^2(AM^{-1})r_1)$$

it may seem like a good idea to compute the sequence $\rho_n^2(AM^{-1})r_1$ since, ideally, this sequence would have double the convergence speed of the original sequence. Here is why computing this sequence is possible.

10.1 Matrix derivation of polynomial squaring methods

Let X be a Krylov sequence of A, that is, $AX = XJ$, and let R consist of linear combination of the X sequence: $R = XU$. From the foregoing discussion we know that we have $AR = RH$ with $H = U^{-1}JU$. For the n-th residual we have $r_n = \rho_n(A)x_1$; see lemma 68 for more details.

We now define in each n-th step a new Krylov vector sequence $Y^{(n)}$ by

$$Y^{(n)} = \rho_n(A)X \Rightarrow AY^{(n)} = Y^{(n)}J.$$

Using again the upper triangular matrix U to take linear combinations, this time from $Y^{(n)}$, we define $S^{(n)}$ by

$$S^{(n)} = Y^{(n)}U \Rightarrow AS^{(n)} = S^{(n)}H. \tag{114}$$

Note that the matrix H is the same as above, and in particular independent of n.

The justification for these new sequences is that

$$y_1^{(n)} = r_n, \qquad s_n^{(n)} = \rho_n(A)y_1^{(n)} = \rho_n^2(A)r_1.$$

Sonneveld [72]'s Conjugate Gradients Squared was the first formulation of a computation of these squared sequences.

There is in fact an unused degree of freedom in the above method: we can choose a different upper triangular matrix V to form the linear combinations $S^{(n)}$, so that

$$S^{(n)} = Y^{(n)}V \Rightarrow AS^{(n)} = S^{(n)}K \tag{115}$$

where $K = V^{-1}JV$ is another Hessenberg matrix. One option would be to let the combinations V correspond to a steepest descent method; the resulting method by van der Vorst [76] is known as BiCG Stabilised.

In all polynomial squaring methods, H is the Hessenberg matrix of the Lanczos method, that is, its elements can be computed from the inner products $s_i^t A r_j$, $s_i^t r_j$ directly or indirectly. For the computation of these inner products the sequence S is not explicitly needed except for its first element. For instance, $s_i^t r_j = s_1 \rho_i(A)\rho_j(A)r_1 = s_1^t r_i^{(j)}$.

10.2 Coefficients in general Polynomial squaring methods

Consider a generalized Lanczos method with right and left residual sequences R and S and upper Hessenberg matrices H and K such that

$$AM^{-1}R = RH \qquad \text{and} \qquad A^tM^{-t}S = SK.$$

For these sequences there are corresponding sequences of polynomials $\{\rho_i\}$ and $\{\sigma_i\}$ so that (see lemma 69) $r_i = \rho_i(AM^{-1})r_1$ and $s_i = \sigma_i(A^tM^{-t})s_1$, where the i-th polynomials have degree $i-1$. The Lanczos method uses the same polynomials for both sequences, so we will introduce a reference sequence $\tilde{R}$ satisfying $A^tM^{-t}\tilde{R} = \tilde{R}H$, $\tilde{r}_1 = s_1$, and when needed reference search directions $\tilde{P}$ satisfying $\tilde{P}U = M^{-t}\tilde{R}$. Equivalently, the elements of these sequences are generated as

$$\tilde{r}_i = \rho_i(A^tM^{-t})\tilde{r}_1.$$

We now consider the matter of computing the elements of H. While we are interested in having for instance the Lanczos coefficient

$$\tilde{r}_i^t M^{-1} r_i = \big(\rho_i(A^tM^{-t})s_1\big)^t M^{-1}\big(\rho_i(AM^{-1})r_1\big) = s_1^t M^{-1}\rho_i^2(AM^{-1})r_1, \tag{116}$$

the only thing we can reasonably compute is

$$\begin{aligned} s_1^t M^{-1} r_i^{(i)} &= s_1^t M^{-1}\sigma_i(AM^{-1})\rho_i(AM^{-1})r_1 \\ &= \big(\sigma_i(A^tM^{-t})s_1\big)^t M^{-1}\big(\rho_i(AM^{-1})r_1\big) \\ &= s_i^t M^{-1} r_i. \end{aligned}$$

The key to retrieving the Lanczos coefficient is in comparing the left sequence $s_i = \sigma_i(A^tM^{-t})s_1$ to the reference Lanczos left sequence $\tilde{r}_i = \rho_i(A^tM^{-t})s_1$, where ρ_i are the polynomials of the right sequence.

Since both are polynomial sequences, with the degrees of corresponding polynomials σ_i and ρ_i the same, there are upper triangular matrices

U and V such that

$$\tilde{R} = KU, \qquad S = KV \qquad \text{where } K = K\langle A^t M^{-t}, s_1\rangle,$$

so the sequences are related by $S = \tilde{R}U^{-1}V$. This relation can be carried over to the coefficients. Below are two ways of using the computed quantities to retrieve the original Lanczos coefficients.

Defining the upper triangular matrix $W = V^{-1}U$, we find that

$$\tilde{R}^t M^{-1} R = W^t S^t M^{-1} R \qquad \text{and} \qquad \tilde{R}^t A R = W^t S^t A R,$$

in which $S^t R$ is lower triangular and $S^t AR$ is of lower Hessenberg form. This implies that we can retrieve the inner products in equations (125) and (126) as

$$\tilde{r}_i^t r_i = w_{ii} s_i^t r_i \qquad \text{and} \qquad \tilde{r}_i^t A r_i = w_{ii} s_i^t A r_i + w_{i-1i} s_{i-1}^t A r_i.$$

The only remaining problem is computing the numbers w_{ii} and w_{i-1i}. Implementations of BiCGstab based on coupled two-term recurrences needs only w_{ii}.

The necessary elements of W follow from the equation $VW = U$:

$$w_{ii} = u_{ii}/v_{ii} \qquad \text{and} \qquad w_{i-1i} = (u_{i-1i} - v_{i-1i} w_{ii})/v_{i-1i-1} \tag{117}$$

where we compute the elements of U from H by the recurrences

$$\begin{aligned} u_{i+1i+1} h_{i+1i} &= u_{ii} \\ u_{ii+1} h_{i+1i} &= u_{i-1i} - u_{ii} h_{ii} \end{aligned}$$

and similar formulas for computing V from K.

Using these formulas directly gives to a method that has a danger of leading quickly to floating point underflow. Therefore we note that we only need the ratios

$$\frac{\tilde{r}_{i+1}^t r_{i+1}}{\tilde{r}_i^t r_i} = \frac{s_{i+1}^t r_{i+1}}{s_i^t r_i} \cdot \frac{k_{i+1i}}{h_{i+1i}} \tag{118}$$

and

$$\frac{\tilde{r}_{i+1}^t A r_{i+1}}{\tilde{r}_{i+1}^t r_{i+1}} = \frac{s_{i+1}^t A r_{i+1}}{s_{i+1}^t r_{i+1}} + \frac{w_{ii+1}}{w_{i+1i+1}} \cdot \frac{s_i^t A r_{i+1}}{s_{i+1}^t r_{i+1}} \tag{119}$$

in order to compute the Hessenberg matrix of the bi-conjugate gradient algorithm; see equation (127).

Similar formulas are arrived at by taking for W the inverse of the above choice.

Define then the upper triangular matrix $W = V^{-1}U$. This leads to

$$S^t R = W^t \tilde{R}^t R \qquad \text{and} \qquad S^t A R = W^t \tilde{R}^t A R.$$

Now we find for the inner products in equations (125) and (126) that

$$w_{ii} \tilde{r}_i^t r_i = s_i^t r_i$$

which leads again to formula (118), and

$$w_{ii} \tilde{r}_i^t A r_i = s_i^t A r_i - w_{i-1i} \tilde{r}_{i-1}^t A r_i.$$

which gives, analogous to by slightly differing from (119)

$$\frac{\tilde{r}_{i+1}^t A r_{i+1}}{\tilde{r}_{i+1}^t r_{i+1}} = \frac{s_{i+1}^t A r_{i+1}}{s_{i+1}^t r_{i+1}} + \frac{w_{ii+1}}{w_{i+1i+1}} \cdot \frac{\tilde{r}_i^t A r_{i+1}}{\tilde{r}_{i+1}^t r_{i+1}}. \tag{120}$$

However, computing according to equations (119) and (120) gives methods that first of all take an extra inner product, and secondly, turn out to be extremely numerically unstable[17]. Therefore we substitute for the first method

$$\frac{w_{ii+1}}{w_{i+1i+1}} = \frac{1}{k_{i+1i}} \left(\frac{u_{ii+1}}{u_{i+1i+1}} - \frac{v_{ii+1}}{v_{i+1i+1}} \right)$$

17. In fact, even on small, well-conditioned problems they stall or diverge within a few iterations.

and interchanging the roles of u and v for the second choice of w

$$\frac{w_{ii+1}}{w_{i+1i+1}} = \frac{1}{h_{i+1i}}\left(\frac{v_{ii+1}}{v_{i+1i+1}} - \frac{u_{ii+1}}{u_{i+1i+1}}\right).$$

Using some elementary relation about the Hessenberg matrices H and K, for instance

$$\tilde{r}_i^t A r_{i+1} = \tilde{r}_{i+1}^t r_{i+1} h_{i+1i}$$

we then find for both choices of W

$$\frac{\tilde{r}_{i+1}^t A r_{i+1}}{\tilde{r}_{i+1}^t r_{i+1}} = \frac{s_{i+1}^t A r_{i+1}}{s_{i+1}^t r_{i+1}} + \frac{u_{ii+1}}{u_{i+1i+1}} - \frac{v_{ii+1}}{v_{i+1i+1}}. \tag{121}$$

The normalized polynomial coefficients u_{ii+1}/u_{i+1i+1}, v_{ii+1}/v_{i+1i+1} can be calculated recursively. For instance,

$$\frac{u_{ii+1}}{u_{i+1i+1}} = \frac{u_{i-1i}}{u_{ii}} - h_{ii}. \tag{122}$$

but in practice we see no difference in numerical stability between using equation (121) directly, and modifying it with (122).

10.3 Standard presentation of Polynomial Squaring methods

We will now present Polynomial Squaring methods based on coupled two-term recurrences.

Denote in preconditioned bi-conjugate gradient-type methods left and right residuals by s_i, r_i and left and right search directions by q_i, p_i, and let M be the preconditioning matrix.

Assume that the left and right sequence are computed from tridiagonal Hessenberg matrices. The residuals and search directions are computed as

$$r_{i+1} = r_i - A p_i \alpha_i^{(r)}, \qquad p_{i+1} = M^{-1} r_{i+1} + p_i \beta_i^{(r)},$$

and the left sequences of residuals and search directions are computed as

$$s_{i+1} = s_i - A^t q_i \alpha_i^{(\ell)}, \qquad q_{i+1} = M^{-t} s_{i+1} + q_i \beta_i^{(\ell)},$$

where $\alpha_i^{(\ell)}$, $\beta_i^{(\ell)}$ and $\alpha_i^{(r)}$, $\beta_i^{(r)}$ may be different for the left and right sequences. The coefficients for the right sequence are computed in the traditional manner

$$\alpha_i^{(r)} = s_i^t M^{-1} r_i / q_i^t A p_i, \qquad \beta_i^{(r)} = s_{i+1}^t M^{-1} r_{i+1} / s_i^t M^{-1} r_i.$$

For the Conjugate Gradient Squared method we choose

$$\alpha_i^{(\ell)} = \alpha_i^{(r)} \equiv \alpha_i, \qquad \beta_i^{(\ell)} = \beta_i^{(r)} \equiv \beta_i.$$

For the BiConjugate Gradient Stabilized we make a different choice, where the left sequence s_i satisfies a two-term recurrence, namely determined by a steepest descent method, giving

$$\alpha_i^{(\ell)} \neq \alpha_i^{(r)}, \beta_i^{(\ell)} = 0.$$

There exist polynomials (see lemma 69) ρ_i, π_i, σ_i, χ_i of degree $i-1$ such that

$$r_i = \rho_i(AM^{-1}) r_1, \quad p_i = M^{-1} \pi_i(AM^{-1}) r_1,$$

and

$$s_i = \sigma_i(A^tM^{-t})s_1, \quad q_i = M^{-t}\chi_i(A^tM^{-t})s_1.$$

We find relations for the polynomials (again, see lemma 69)

$$\rho_{i+1}(x) = \rho_i(x) - x\pi_i(x)\alpha_i, \qquad \pi_{i+1} = \rho_{i+1} + \pi_i\beta_i.$$

For cgs the polynomials for the left and right sequences are identical. In bcgs we base the left sequence on the steepest descent method. This implies that the search directions are identical to the residuals, and for the left polynomials we find

$$\sigma_{i+1}(x) = \sigma_i(x) - x\chi_i\alpha_i^{(\ell)}, \qquad \chi_{i+1} = \sigma_{i+1}. \tag{123}$$

10.3.1 Vector recurrences

The derivation of the squared residuals $\sigma_i(M^{-1}A)M^{-1}r_i$, and various auxiliaries, makes extensive use of the relations of equation (145) between polynomials. For compactness of the exposition we will mostly omit the polynomial argument AM^{-1}, that is, if we write ρ_i without an argument, an argument of AM^{-1} is implicitly assumed.

First a strictly auxiliary quantity:

$$\begin{aligned}
\boxed{1}_{i+1} &\stackrel{D}{=} \chi_i(M^{-1}A)M^{-1}r_{i+1} \\
&= \chi_i(M^{-1}A)M^{-1}r_i - \chi_i(M^{-1}A)M^{-1}Ap_i\alpha_i^{(r)} \\
&= \chi_i(M^{-1}A)M^{-1}r_i - M^{-1}A\chi_i(M^{-1}A)p_i\alpha_i^{(r)} \\
&= \begin{cases} \boxed{3}_i - M^{-1}A\boxed{4}_i\alpha_i^{(r)} & CGS \\ \boxed{2}_i - M^{-1}A\boxed{4}_i\alpha_i^{(r)} & BiCGstab \end{cases}
\end{aligned}$$

Next the squared residual:

$$\begin{aligned}
\boxed{2}_{i+1} &\stackrel{D}{=} \sigma_{i+1}(M^{-1}A)M^{-1}r_{i+1} \\
&= (\sigma_i(M^{-1}A) - M^{-1}A\chi_i(M^{-1}A)\alpha_i^{(\ell)})M^{-1}r_{i+1} \\
&= \sigma_i(M^{-1}A)M^{-1}r_i - \sigma_i(M^{-1}A)M^{-1}Ap_i\alpha_i^{(r)} \\
&\qquad - M^{-1}A\chi_i(M^{-1}A)M^{-1}r_{i+1}\alpha_i^{(\ell)} \\
&= \boxed{2}_i - M^{-1}A\sigma_i(M^{-1}A)p_i\alpha_i^{(r)} - M^{-1}A\boxed{1}_{i+1}\alpha_i^{(\ell)} \\
&= \begin{cases} \boxed{2}_i - M^{-1}A(\boxed{3}_i\alpha_i^{(r)} + \boxed{1}_{i+1}\alpha_i^{(\ell)}) & CGS \\ \boxed{2}_i - M^{-1}A(\boxed{4}_i\alpha_i^{(r)} + \boxed{1}_{i+1}\alpha_i^{(\ell)}) & BiCGstab \end{cases}
\end{aligned}$$

where we used that

$$\begin{aligned}
&\sigma_i(M^{-1}A)p_i = \sigma_i(M^{-1}A)M^{-1}\pi_i r_1 \\
&= \begin{cases} M^{-1}\sigma_i\pi_i r_1 = M^{-1}\pi_i\sigma_i r_1 = \pi_i(M^{-1}A)M^{-1}\sigma_i r_1 & \\ \qquad = \pi_i(M^{-1}A)M^{-1}r_i = \boxed{3}_i & CGS \\ \chi_i(M^{-1}A)p_i = \boxed{4}_i & BiCGstab \end{cases}
\end{aligned}$$

The next quantity is not needed for BiCGstab, since by $\chi_i \equiv \sigma_i$ it is equal to the previous:

$$\begin{aligned}
\boxed{3}_{i+1} &\stackrel{D}{=} \chi_{i+1}(M^{-1}A)M^{-1}r_{i+1} \\
&= \sigma_{i+1}(M^{-1}A)M^{-1}r_{i+1} + \chi_i(M^{-1}A)M^{-1}r_{i+1}\beta_i^{(\ell)} \\
&= \boxed{2}_{i+1} + \boxed{1}_{i+1}\beta_i^{(\ell)} \qquad \text{CGS only}
\end{aligned}$$

For the squared search directions we find

$$\boxed{4}_{i+1} \stackrel{D}{=} \chi_{i+1}(M^{-1}A)p_{i+1}$$
$$= \begin{cases} \sigma_{i+1}(M^{-1}A)p_{i+1} + \chi_i(M^{-1}A)p_{i+1}\beta_i^{(\ell)} & CGS \\ \sigma_{i+1}(M^{-1}A)p_{i+1} & BiCGstab \end{cases}$$

(now, for CGS use

$$\begin{aligned} \sigma_{i+1}(M^{-1}A)p_{i+1} &= M^{-1}\sigma_{i+1}\pi_{i+1}r_1 = M^{-1}\pi_{i+1}\sigma_{i+1}r_1 \\ &= \pi_{i+1}(M^{-1}A)M^{-1}r_{i+1} = \boxed{3}_{i+1}; \end{aligned}$$

for BiCGstab use

$$\begin{aligned} \sigma_{i+1}(M^{-1}A)p_{i+1} &= \sigma_{i+1}(M^{-1}A)(M^{-1}r_{i+1} + p_i\beta_i^{(r)}) \\ &= \boxed{1}_{i+1} + (\sigma_i(M^{-1}A) + M^{-1}A\chi_i(M^{-1}A)\alpha_i^{(\ell)})p_i\beta_i^{(r)} \\ &= \boxed{1}_{i+1} + (\chi_i(M^{-1}A) + M^{-1}A\chi_i(M^{-1}A)\alpha_i^{(\ell)})p_i\beta_i^{(r)} \end{aligned}$$

to find:)

$$\boxed{4}_{i+1} = \begin{cases} \boxed{3}_{i+1} + \chi_i(M^{-1}A)(M^{-1}r_{i+1} + p_i\beta_i^{(r)})\beta_i^{(\ell)} \\ \quad = \boxed{3}_{i+1} + (\boxed{1}_{i+1} + \boxed{4}_i\beta_i^{(r)})\beta_i^{(\ell)} & CGS \\ \boxed{1}_{i+1} + (\boxed{4}_i + M^{-1}A\boxed{4}_i\alpha_i^{(\ell)})\beta_i^{(r)} & BiCGstab \end{cases}$$

In order to start the algorithm off, we note that from $\rho_1(M^{-1}A)r_1 = r_1$:

$$\rho_1(x) = \sigma_1(x) = \pi_1(x) = \chi_1(x) = 1,$$

giving

$$\boxed{2}_1 = \boxed{3}_1 = \boxed{4}_1 = M^{-1}r_1,$$

and we enter the iteration loop by computing $\boxed{1}_{i+1}$ for $i = 1$.

Question: *Derive the solution update.*

Question: *Derive error estimate in each iteration.*

10.3.2 Work estimates

The amount of work per iteration for both methods consists of two matrix-vector products and preconditioner solves, performed in the computation of $\boxed{1}_{i+1}$ and $\boxed{2}_{i+1}$. In BiCGstab, the vector $M^{-1}A\boxed{4}_i$, computed for the construction of $\boxed{1}_{i+1}$, can be reused in the construction of $\boxed{4}_{i+1}$. Additionally, for cgs there are 6 vector additions or vector-plus-scalar-times-vector operations, for bcgs there are 4 such operations. Updating the iterate takes one extra vector operation (analogous to updating $\boxed{2}_i$) for cgs and two (corresponding to update $\boxed{1}_i$ and $\boxed{2}_i$) for bcgs.

10.3.3 Computation of coefficients in CGS

In CGS, we take the same update for the left and right sequence, so first of all $\alpha_i^{(\ell)} \equiv \alpha_i^{(r)}$ and $\beta_i^{(\ell)} \equiv \beta_i^{(r)}$.

For the computation of the α and β scalars we need the $\tilde{r}_i^t M^{-1}r_i$ and $\tilde{p}_i^t Ap_i$ inner products. Using the vectors derived above,

$$\tilde{r}_i^t M^{-1}r_i = (\sigma_i(A^tM^{-t})\tilde{r}_1)^t M^{-1}r_1 = \tilde{r}_1^t\sigma_i(M^{-1}A)M^{-1}r_i = \tilde{r}_1^t\boxed{2}_i,$$

and

$$\tilde{p}_i^t Ap_i = \tilde{r}_1^t\chi_i(M^{-1}A)M^{-1}Ap_i = \tilde{r}_1^t M^{-1}A\tilde{\pi}_i(M^{-1}A)p_i = \tilde{r}_1 M^{-1}A\boxed{4}_i.$$

Note that $M^{-1}A\boxed{4}_i$ is computed in the above algorithm.

10.3.4 Full CGS algorithm

We give the full CGS algorithm in algorithm 7.

Algorithm: Conjugate Gradients Squared

Algorithm 7 *Let A, M, and x_1 be given. Compute $r_1 = Ax_1 - b$ and $p_1 = M^{-1}r_1$. Now iterate for $i = 1, \ldots$:*

- *Compute one dot product $\rho_i = r_1^t M^{-1} r_i$, and, if $i > 1$, $\beta_i = \rho_i/\rho_{i-1}$.*
- *If $i > 1$, compute a temporary $\boxed{3}_i$:*
$$s = r_i + q_{i-1}\beta_i.$$
- *Update the squared search directions $\boxed{4}_i$ for $i > 1$:*
$$p_i = s + (q_i + p_i\beta_i)\beta_i.$$
- *Compute the product $M^{-1}Ap_i$ with the search directions, and use it for the scalars*
$$\pi_i = r_1^t M^{-1} A p_i, \qquad \alpha_i = \rho_i/\pi_i.$$
- *Update the auxiliary quantity $\boxed{1}_i$, using the product $M^{-1}Ap_i$ already computed:*
$$q_i = p_i - M^{-1}Ap_i\alpha_i.$$
- *Compute the update vector*
$$t = (p_i + q_i)\alpha_i,$$
and use it to update the solution
$$x_{i+1} = x_i - t,$$
and residual $\boxed{2}_{i+1}$:
$$r_{i+1} = r_i - M^{-1}A * t,$$
where for the residual update we perform the second multiplication by $M^{-1}A$.

We give the algorithm as Matlab code in section A.2.

10.3.5 Computation of coefficients in BiCGstab

10.4 Three-term recurrence derivation Conjugate Gradient Squared

For expository purposes it is most convenient to derive first the version of Conjugate Gradients Squared based on three-term recurrences, and without preconitioning. We present the practically preferred versions of CGS and BiCGstab, based on coupled two-term recurrences, and with preconditioning, in section 10.3.

Naturally, we do not wish to build the whole sequence $S^{(n)}$ (see equation (115)) in the n-th step. It turns out that of each $S^{(n)}$ we need only four elements[18].

The residual polynomials π_n are related by

$$\pi_{n+1}(t)h_{n+1n} + \pi_n(t)h_{nn} + \pi_{n-1}(t)h_{n-1n} = t\pi_n(t); \tag{124}$$

(see lemma 68). From this we find that

$$S^{(n+1)}h_{n+1n} + S^{(n)}h_{nn} + S^{(n-1)}h_{n-1n} = AS^{(n)}$$

and combining this with (114) we find the following computation:

$$\begin{aligned}
s_{n-1}^{(n+1)}h_{n+1n} &= As_{n-1}^{(n)} - (s_{n-1}^{(n)}h_{nn} + s_{n-1}^{(n-1)}h_{n-1n}) \\
s_{n}^{(n+1)}h_{n+1n} &= As_{n}^{(n)} - (s_{n}^{(n)}h_{nn} + s_{n}^{(n-1)}h_{n-1n}) \\
s_{n+1}^{(n+1)}h_{n+1n} &= As_{n}^{(n+1)} - (s_{n}^{(n+1)}h_{nn} + s_{n-1}^{(n+1)}h_{n-1n}) \\
s_{n+2}^{(n+1)}h_{n+2n+1} &= As_{n+1}^{(n+1)} - (s_{n+1}^{(n+1)}h_{n+1n+1} + s_{n}^{(n+1)}h_{nn+1})
\end{aligned}$$

where the left hand sides are computed in the order stated. In the last two steps we perform a matrix-vector product; the results of these can be reused in the first two steps of the next iteration. We see that this method requires two matrix-vector products, but neither is with the transpose of the matrix.

18. This presentation is in the context of a tridiagonal matrix H generating the iterative method to be squared, as is the case with BiCG. More general methods may be possible.

For the computation of the coefficients of H we have to go back to the Lanczos method. In section 8.3 we saw that we need the expressions (now denoting the left sequence of residuals by $\tilde{r}_n$)

$$\tilde{r}_n^t A r_n \quad (74), \qquad \tilde{r}_n^t r_n \quad (74) \text{ and } (75).$$

From the above it is easy to see that

$$\tilde{r}_n^t r_n = \tilde{r}_1^t \pi_n^2(A) r_1 = \tilde{r}_1^t s_n^{(n)}, \qquad \tilde{r}_n^t A r_n = \tilde{r}_1^t A \pi_n^2(A) r_1 = \tilde{r}_1^t A s_n^{(n)}.$$

Thus the left sequence can be eliminated completely, and only its first element is ever needed. Using the above two expression, the elements of H can be computed, for instance as in equation (78).

10.4.1 Computation of coefficients in CGS

In the conjugate gradient squared method, the choices $\rho_i \equiv \sigma_i$ and $H = K$ are made, where the tridiagonal matrix H is chosen as the one generated by the Lanczos method. Denoting the left and right residuals of the Lanczos method by s_i, r_i, we need the values of

$$s_i^t r_i = \big(\rho_i(A^t) r_1\big)^t \big(\rho_i(A) r_1\big) = r_1^t \rho_i^2(A) r_1 = r_1^t r_i^{(i)}, \tag{125}$$

and

$$s_i^t A r_i = \big(\rho_i(A^t) r_1\big)^t A \big(\rho_i(A) r_1\big) = r_1^t A \rho_i^2(A) r_1 = r_1^t A r_i^{(i)}. \tag{126}$$

The coefficients of H then follow from

$$h_{ii} = \frac{s_i A r_i}{s_i^t r_i}, \qquad h_{i-1i} = \frac{s_i r_i}{s_{i-1}^t r_{i-1}} h_{ii-1}, \qquad h_{i+1i} = -h_{ii} - h_{i-1i}. \tag{127}$$

(For a survey of such identities, see [24].)

10.4.2 Computation of coefficients in BiCGstab

The BiCGstab algorithm is based on taking a different recurrence in i direction from the one in j direction. While the latter one is still based on the matrix H from the Lanczos method, the former recurrence takes steepest descent steps. Thus, we perform the steepest descent update

$$r_{i+1}^{(i+1)} = r_i^{(i+1)} + \omega_i A r_i^{(i+1)}$$

with

$$\omega_i = -\frac{r_i^{(i+1)^t} A r_i^{(i+1)}}{(A r_i^{(i+1)})^t (A r_i^{(i+1)})}.$$

In order to write this as a three-term recurrence

$$r_{i+1}^{(i+1)} k_{i+1i} + r_i^{(i+1)} k_{ii} + r_{i-1}^{(i+1)} k_{i-1i} = A r_i^{(i+1)}$$

we choose $k_{i-1i} = 0$, and

$$k_{i+1i} = -k_{ii} = 1/\omega_i.$$

11 Arnoldi methods

Previous section: 10 Next section: 12

In this monograph we have repeatedly considered the Arnoldi orthogonalisation, that is, generating a sequence R that is orthogonal to itself under some inner product. Usually we considered the case where R was a residual sequence.

There is another application for the Arnoldi method. If

$$AV = VH \qquad \text{and} \qquad V^tV = I$$

then

$$V^tAV = H \tag{128}$$

is a similarity transformation to upper Hessenberg (or in case of symmetry, tridiagonal) form. Many eigenvalue methods perform this reduction as a preliminary step.

In this section we will investigate what is usually termed the 'Arnoldi method', which is the above similarity transformation. Methods based on Arnoldi orthogonalisation that generate residual sequences are considered in section 9.

11.1 The Arnoldi algorithm

Here we will give the algorithm for the Arnoldi similarity transformation equation (128); we will immediately formulate it in block form, where each vector is of size $n \times b$.

Algorithm: Arnoldi reduction to Hessenberg form

Algorithm 8 *Let a matrix A and a vector v_1 be given.*

1. *Assume inductively that*

$$v_i^tv_i = I_{b\times b} \qquad \text{and} \qquad \forall_{j<i}\colon v_j^tv_i = 0_{b\times b}.$$

2. *Let $u \leftarrow Av_i$.*
3. *For all $j \leq i$ modify*

$$u \leftarrow u - v_j(v_j^tu)^{-1}.$$

4. *Let w be square of size $b \times b$ such that $u^tu = (ww^t)^{-1}$, and define*

$$v_{i+1} \leftarrow uw.$$

In sum,

$$v_{i+1}w = Av_i - \sum_{j\leq i} v_j(v_j^tu)^{-1} \quad \Rightarrow \quad Av_i = v_{i+1}w + \sum_{j\leq i} v_j(v_j^tu)^{-1},$$

which gives us the coefficients of the Hessenberg matrix relating the various quantities in $AV = VH$.

One easily sees that step 3 guarantees that v_{i+1} is inductively orthogonal to all v_j with $j \leq i$; step 4 guarantees that $v_{i+1}^tv_{i+1} = I_{b\times b}$. In the scalar case, w is simply $\pm 1/\sqrt{u^tu}$; in the block case we make a QR decomposition $u = qr$, and choose $w = r^{-1}$. QR decompositions are uniquely determined up a diagonal matrix D satisfying $D^2 = I$.

This process is needed in the GMRES algorithm (section 9.5). We can analyze its breakdown as follows:

- The orthogonalisation in step 3 can be omitted if $v_j^tu = 0$, so this step can not break down.
- In step 4 the occurrence of $ww^t = 0$ implies that we have found an invariant subspace, so by corollary 22 we can form the solution to the linear system.

11.2 Characterisation of the Arnoldi sequence

Based on a Krylov sequence K, many sequences can be formed by taking combinations $R = KU$. The sequences are completely characterised by the upper triangular matrix U; for the subset of the residual sequences the only restriction on U is that $u_{1j} \equiv 1$. We now address the question how to characterise the Arnoldi sequence – which is not a residual sequence – in terms of its matrix U. Since the Arnoldi sequence is uniquely determined (up to signs) from the first vector (section 11.1), there has to be a matrix U, unique up to sign scaling, derivable from the Krylov sequence.

The determining characteristic of the Arnoldi sequence is that it satisfies $V^tV = I$. Since the Arnoldi sequence is a combination $V = KU$ of the Krylov sequence, we have

$$U^tK^tKU = I \quad \Rightarrow \quad U^{-t}U^{-1} = K^tK,$$

that is, U is the inverse Cholesky factor of the moments matrix K^tK.

For the Arnoldi sequence, $K = VU^{-1}$ is a QR factorisation of the Krylov sequence. This is unique up to scaling as $V \leftarrow VD$ and $U^{-1} \leftarrow DU^{-1}$ by a diagonal matrix D with $D^2 = I$. In $K^tK = U^{-t}U^{-1}$ this diagonal scaling drops out.

The Hessenberg matrix H in $AV = VH$ is $H = U^{-1}JU$. This makes H determined up to scaling DHD. In particular, the diagonal of H is invariant.

Uniqueness of Arnoldi method

Lemma 63 *If $AV_1 = V_1H_1$ with $V_1^tV_1 = I$ and $AV_2 = V_2H_2$ with $V_2^tV_2 = I$, then there is a diagonal matrix D with $D^2 = I$ such that*

$$V_1 = V_2D, \qquad H_1 = DH_2D.$$

Alternative deriviation. The Arnoldi sequence V satisfies an equation $AV = VH$ just like all sequences that are derived by taking combinations of a Krylov sequence; see lemma 29. From this, we find $V^tAV = H$. Now use equation (31): $H = U^{-1}JU$, where $V = KU$, to find

$$V^tAV = \begin{cases} H = U^{-1}JU \\ U^tK^tAKU = U^tK^tKJU \end{cases}$$

from which

$$K^tK = U^{-t}U^{-1}.$$

This requires a little care. We can only partly 'divide out' the singular matrix JU. At first it gives us that the result holds from the second column onward. Because of symmetry it then also holds from the second row down. And now for the $(1,1)$ component...

11.3 Orthogonalisation of the Krylov sequence

Iterative methods based on Arnoldi orthogonalisation perform a substantial number of orthogonalisation steps in each iteration. The Blas1 operations associated with this are quite inefficient on most computers. We will discuss two strategies of improving the performance of Arnoldi-based methods.

Apart from the Gram-Schmidt-based strategies below, there is also the possibility of using Householder reflections [82].

11.3.1 Gram-Schmidt and modified Gram-Schmidt in Arnoldi methods

The orthogonalisation in each iteration of GMRES or OrthoDir can be desribed as in figure 4, which is a Modified Gram-Schmidt process. The

---

$w \leftarrow Av_i$
for $j < i$
  let $\delta_j = (w, v_j)$
  update $w \leftarrow w - \delta_j v_j$.
$v_{i+1} = w/\|w\|$

Figure 4: Modified Gram-Schmidt orthogonalisation

---

inner products and vector updates performed here are all Blas1 operation, hence relatively inefficient on most computers. Also, their interdependence implies a large number of synchronisation points in a parallel context

An easy way to increase the efficiency of the orthogonalisation process is to use a classical Gram-Schmidt process as in figure 5. Performing the inner products and updates in a block manner raises the execution efficiency to Blas2 level, and all inner products can be combined in a single

---

$w \leftarrow Av_i$
for $j < i$
  let $\delta_j = (w, v_j)$
for $j < i$
  update $w \leftarrow w - \delta_j v_j$.
$v_{i+1} = w/\|w\|$

Figure 5: Classical Gram-Schmidt orthogonalisation

---

global reduction operation. Unfortunately, this method is less numerically stable; see [9, 10, 18].

Question: *How do the Hessenberg matrices relate for the two variants?*

A solution, proposed in [18], is to re-orthogonalise in case of proven large numerical errors; see figure 6. The tolerance parameter η to base the re-

---

$w \leftarrow Av_i$
`label:ortho`
for $j < i$
  let $\delta_j = (w, v_j)$
for $j < i$
  update $w \leftarrow w - \delta_j v_j$.
if $\|w\| < \eta\|(\ldots \delta_j \ldots)\|$
  repeat once from label `ortho`.
$v_{i+1} = w/\|w\|$

Figure 6: Classical Gram-Schmidt orthogonalisation

---

orthogonalization decision on can be fairly modest, e.g., $1/\sqrt{2}$. The η test tests for the occurence of cancellation.

Since the coefficients arising here have to be stored in a Hessenberg matrix, we derive the effect of this double orthogonalisation. Let $v_i, i = 1 \ldots n$ be orthogonal, and let $u = Av_n$. Then:

$$\begin{aligned} u' &= u - \sum_i^n \alpha_i v_i, \quad \alpha_i = \frac{v_i^t u}{v_i^t v_i} v_i \\ u'' &= u' - \sum_i^n \beta_i v_i, \quad \beta_i = \frac{v_i^t u'}{v_i^t v_i} v_i \\ &= u - \sum_i^n (\alpha_i + \beta_i) v_i \end{aligned}$$

After scaling $v_{n+1} = u''/\gamma_n$ (we leave open the possibility that the v_i vectors are not normalised), we get

$$Av_n = v_{n+1}\gamma_n + \sum_i^n (\alpha_i + \beta_i) v_i.$$

An approach in between Classical and Modified Gram-Schmidt is the Block Gram-Schmidt algorithm [43]; figure 7. Here all vectors are block

---

$W \leftarrow AV_i$
for $j < i$
  $\Delta_j = V_i^t W$
  $W \leftarrow W - V_i \Delta$

Figure 7: Block Gram-Schmidt orthogonalisation

---

vectors. If V_i is a true block vector, this algorithm uses Blas3 operations, otherwise the inner products and updates are still of Blas2 level.

Vanderstraeten [79] proposes to let the size of the V_i blocks be dynamically determined by their condition, which can be efficiently estimated from the R matrices of their QR factorisations; [8]. The algorithm is given in figure 8.

$w \leftarrow Av_i$
for $j < p$
  orthogonalise w again block V_j
$V_p \leftarrow [V_p, w]$
if the condition of V_p is too large,
  start building a new V block,
  otherwise, continue adding to this block

Figure 8: Dynamic Block Gram-Schmidt orthogonalisation

11.3.2 After-the-fact orthogonalisation

All algorithms discussed in this monograph alternate the matrix-vector product and the orthogonalisation of the result to the previously generated vectors. It is possible to generate an initial part K_n of the Krylov sequence and orthogonalise this as a whole, using a QR decomposition. Writing this decomposition as

$$K_n = V_n U_n^{-1}$$

we thus derive an orthonormal V_n, which can be scaled to a residual sequence (section 16.11), or used in a GMRES method (section 9.5). The Hessenberg matrix can be constructed as $H = U^{-1}JU$; see equation (31).

This procedure is not recommended for any large values of n, since roundoff will steer the k_i vectors towards the largest eigenvector. However, for moderate values there may be a performance advantage to this approach:

- The construction of U_n is now done as a whole, and not by updating, column by column. Thus, a Blas 3 kernel can be used for the QR decomposition. Experience shows that, because of register blocking, already for $n = 4$ Blas3 routines may outperform Blas2 ones.
- In parallel it is possible to pipeline the matrix-vector products.

For this, the processors exchange enough boundary information that several products can be performed by purely local operations. This procedure does involve some redundant operations. See [**?**] for details.

- There are other considerations of cache reuse that make this algorithm attractive; see [14].

Instead of deriving the Krylov sequence itself, one could use an inner product-free method such as Chebyshev semi-iteration prior to the block orthogonalisation. This was proposed by de Sturler [21].

11.4 Reduction to Hessenberg form by Householder reflections

For further reading: the material in this section is not referred to elsewhere.

In this monograph we extensively studied the reduction of a matrix to Hessenberg form by forming the Lanczos basis. Such a reduction can also be effected by Housholder reflections. The two reductions are related, and in this section we study precisely how.

The matlab code for various algorithms is in section A.4.

11.4.1 Basic ideas of Householder reflectors

Let u be a normalised vector, and define the Householder reflector $Q_u = I - 2uu^t$ which has the properties

$$Q_u u = -u, \qquad v \perp u \Rightarrow Q_u v = v, \qquad Q_u = Q_u^t = Q_u^{-1}.$$

Now let v and w be vectors with the same norm, and let $u = (v-w)/\|v-w\|$, then Q_u has the properties

$$Q_u v = w, \qquad Q_u w = v, \qquad z \perp v - w \Rightarrow Q_u z = z.$$

Proof: write $v = (v+w)/2 + (v-w)/2$; $Qv = Q(v+w)/2 + Q(v-w)/2 = (v+w)/2 - (v-w)/2 = w$.

We can now effect a reduction $AV = VH$ of an arbitrary A to upper Hessenberg form as follows. Let

$$a \equiv \begin{pmatrix} 0 \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{pmatrix}, \qquad b = \|a\| \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

then, with $u = (a-b)/\|a-b\|$[19],

$$Q_u A = \begin{pmatrix} a_{11} & * & \cdots & * \\ * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix}, \qquad Q_u A Q_u = \begin{pmatrix} a_{11} & * & \cdots & * \\ * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix}.$$

In the symmetric case, $Q_u A Q_u$ will also have zeros in the top row, corresponding to those in the first column.

Now call $Q_1 \equiv Q_u$, and repeat this story on the remaining matrix $A_2 \equiv Q_1 A Q_1$ with, in terms of elements of A_2,

$$a \equiv \begin{pmatrix} 0 \\ 0 \\ a_{32} \\ a_{42} \\ \vdots \\ a_{n2} \end{pmatrix}, \qquad b = \|a\| \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

In this manner we get a reduction to Hessenberg form:

$$Q_n \cdots Q_1 A Q_1 \cdots Q_n = H \qquad \Rightarrow \qquad AQ = QH \text{ with } Q = Q_1 \cdots Q_n.$$

In the symmetric case this is a reduction to tridiagonal form. We note that the matrix Q has the special form

$$Q = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix} \tag{129}$$

19. If a and b are already close, we expect a large roundoff error. Hence it may be preferable to choose $b = -\text{sign}(a_{21})\|a\|e_2$. For more on roundoff analysis of Householder reflectors, see Golub and Van Loan [38], and Wilkinson [86].

11.4.2 Relation to Lanczos vectors

We now study the relation between the above reduction to Hessenberg form by Householder reduction, and the reduction by forming an Arnoldi basis as studied in most of this monograph.

First of all, we note from equation (129) that in the reduction $AV = VH$ by Householder reflectors $v_1 = e_1$, so this reduction is equivalent to applying an Arnoldi process with the first unit vector as starting point.

Conversely, let u and v be normalised vectors and Q a Householder reflector with $v = Qu$, and define $\hat{A} = QAQ$, which is a similarity transformation, then

$$Av = AQu = Q\hat{A}u, \quad \ldots \quad , A^n v = Q\hat{A}^n u$$

and

$$Au = Q\hat{A}Qu = Q\hat{A}v, \quad \ldots \quad , A^n u = Q\hat{A}^n v$$

In other words, the Krylov space $K = K\langle A, v\rangle$ can be found by applying a reflector to $\hat{K} = K\langle \hat{A}, u\rangle$, and vice versa.

Applying this converse statement to the Lanczos starting vector q_1, we note that

$$Q^t A^k q_1 = (Q^t A Q)^k Q^t q_1 = H^k e_1,$$

so

$$Q^t K\langle A, q_1\rangle = K\langle H, e_1\rangle.$$

Another relation statement. Suppose $AQ = QH$ is any reduction to Hessenberg form by an orthonormal Q, for instance as a Lanczos basis. The Q matrix obtained from the reduction by Householder reflectors has the special form of equation (129), so let Q_e be a reflector that transforms q_1 into e_1. Define $\hat{Q} = Q_e Q$, which is of the desired form, and which is orthonormal again. Then

$$AQ = QS \quad \Rightarrow \quad Q_e A Q_e \hat{Q} = \hat{Q} H.$$

It would have been more interesting if we could have transformed the Lanczos basis reduction of A into a Householder reflector basis of A. However, this would imply that after solving one linear system we could solve any other system by a simple transformation of the Lanczos basis. It is unlikely that this can be done.

11.4.3 Retrieving Householder vectors from the Lanczos basis

We will now consider the problem of reconstructing the Householder vectors from the Lanczos basis.

In our first approach we assume that the whole Lanczos basis is available.

Let Q be an orthonormal matrix of the form (129). From the observation that the second column is $e_2 - u_{11}u_1$, where u_1 is the first Householder vector, we can easily compute u_1. Since $Q = Q_1Q_2\cdots$ and $Q_i = Q_i^{-1} = Q_i^t$, we get $Q_2Q_3\cdots = Q_1Q$. Now we can iterate this process to find the second Householder vector, u_2, and eliminate Q_2 from the product.

It is also possible to reconstruct the Householder vectors incrementally.

Let

$$Q = [q_1, \ldots, q_n]$$

be the Lanczos basis. For Q we also have $Q = \prod Q_i$ where $Q_i = I - u_i u_i^t$. Summarise the u_i vectors in

$$U = [u_1, \ldots, u_n]$$

which is strictly lower triangular. The Q_i matrices are of the form

$$Q_i = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & * & \cdots & * \\ & & & \vdots & & \vdots \\ & & & * & \cdots & * \end{pmatrix}$$

The first Householder vector, u_1, can be found from the second column of Q_1, which stays intact in the final matrix Q.

For the second Householder vector we consider the third column of Q, which is also the third column of Q_1Q_2. It is formed from the third column of Q_2, which is $-u_2u_{22} + e_3$, and the rows of Q_1, which are

$$(Q_1)_{k,*} = -u_{k1}u_1^t + e_k^t.$$

As a result we get

$$\begin{aligned} q_3 &= (-u_{*1}u_1^t + e*^t)(-u_2u_{22} + e_3) \\ &= u_{*1}u_1^t u_2 u_{22} - e_*^t u_2 u_{22} - u_{*1}u_1^t e_3 + e_*^t e_3 \\ &= u_{*1}(u_1^t u_2 u_{22} - u_{13}) - u_{*2}u_{22} + \delta_{*3}. \end{aligned}$$

In general we find $q_{n+1} = \sum_k^n u_{*k}c_{kn} + e_{n+1}$, which obviously can be inverted to

$$q_{n+1} = \sum_k^n q_{*k}c_{kn} + u_n\alpha_n + e_{n+1}.$$

This looks suspiciously like the Lanczos update formula.

11.5 Implicitly restarted Arnoldi

Let $AV_m = V_{m+1}H_{[m+1,m]} = V_mH_m + f_me_m^t$ and $m = k+p$. Find the eigenvalues of H and perform $p = m-k$ steps of QR with the p wanted eigenvalues as shifts. This gives $Q = Q_1 \cdots Q_p$. From the observation that $e_i^tQ_j \in [e_{i-1}, e_i, \ldots]$, it follows that $e_m^tQ = \alpha e_k + \cdots$.

Define $\tilde{V}_m = V_mQ$, $\tilde{H}_m = Q^*H_mQ$, then

$$\begin{aligned} A\tilde{V}_m &= \tilde{V}_m\tilde{H}_m + f_me_m^tQ \\ &= \tilde{V}_m\tilde{H}_m + \tilde{f}_ke_k^t + \tilde{f}_{k+1}e_{k+1}^t + \cdots \end{aligned}$$

so the first k columns satisfy an Arnoldi relation:

$$A\tilde{V}_k = \tilde{V}_k\tilde{H}_k + \tilde{f}_ke_k^t$$

This can be used as the starting point of p further iterations.

12 Non orthogonality-based methods

Previous section: 11 Next section: 13

For further reading: the material in this section is not referred to elsewhere.

There are various methods that do not base their workings on some form or orthogonality. In this section we discuss

- Stationary iteration, which reduces the residual in norm by a fixed factor in each iteration.
- Steepest descent, which in each iteration minimizes the residual along the search direction, which is taken as the gradient.
- Chebyshev semi-iteration, which achieves the maximal reduction of the residual in each iteration, based on an estimate of the spectrum.

12.1 Iterative refinement

Given a system $A\bar{x} = f$ and $M \approx A$, set $x_1 = M^{-1}f$, and define $r_1 = Ax_1 - f$ and $e_1 = x_1 - \bar{x} = A^{-1}r_1$, since $\bar{x} = x_1 - A^{-1}r_1$, we iterate $x_2 = x_1 - M^{-1}r_1$.

The terms iterative refinement is also used in the context of direct methods, where a system is solved a small number of times, without further parameters in the correction since the full matrix solve is close to the exact solve.

12.1.1 Propagation of error and residual

For the error $e_n = x_n - x$ (where x is the exact solution of $Ax = f$) and the residual $r_n = Ax_n - b$, we have $Ae_n = r_n$, so

$$e_{n+1} - e_n = -M^{-1}Ae_n \Rightarrow e_{n+1} = (I - M^{-1}A)e_n = Ge_n \tag{130}$$

and

$$r_{n+1} - r_n = AM^{-1}Ae_n = AM^{-1}r_n \Rightarrow r_{n+1} = (I - AM^{-1})r_n \tag{131}$$

The operators are similar:

$$I - AM^{-1} = M(I - M^{-1}A)M^{-1} = A(I - M^{-1}A)M^{-1}.$$

12.2 Rewrites of Stationary Iteration

The basic equation for polynomial iterative methods is

$$x_{n+1} - x_1 \in \text{Span}\{M^{-1}r_1, \ldots, M^{-1}r_n\}$$

where M is a preconditioner matrix, or equivalently,

$$x_{n+1} - x_n \in \text{Span}\{M^{-1}r_1, \ldots, M^{-1}r_n\}.$$

For stationary iteration this becomes

$$x_{n+1} - x_n = -M^{-1}r_n.$$

Equivalently, with the splitting $A = M - N$ and $G = I - M^{-1}A = M^{-1}N$ the error propagation operator (equation 130),

$$x_{n+1} = Gx_n + (I - G)A^{-1}b = Gx_n + M^{-1}b.$$

The update equation $x_{n+1} - x_n = -M^{-1}r_n$ is often written as $M(x_{n+1} - x_n) = -rn$. Rearranging the terms in this gives

$$\begin{aligned} Mx_{n+1} &= Mx_n - r_n \\ &= Ax_n - Nx_n - (Ax_n - b) \\ &= Nx_n + b \end{aligned}$$

To turn this around: the scheme $Mx_{n+1} = Nx_n + b$ is an iterative method for the system $(M - N)x = b$. As a corollary, for any B

$$(M + B)x_{n+1} = (N + B)x_n - b$$

is also a method for $(M - N)x = b$.

If the splitting involves a scaling, for instance in SOR

$$(\omega^{-1}D + L)x_{n+1} = (\omega^{-1}D + U)x_n + b$$

this is also written as

$$(D + \omega L)x_{n+1} = (D + \omega U)x_n + \omega b.$$

12.3 Stationary Iteration with varying solver

Let varying splittings of the coefficient matrix be given, say

$$A = M_1 - N_1 = M_2 - N_2 = M_3 - N_3 = \dots$$

and let iterates be generated by

$$\begin{aligned} x_{i+1} &= x_i - M_1^{-1} r_i \\ x_{i+2} &= x_{i+1} - M_2^{-1} r_{i+1} \\ x_{i+3} &= x_{i+2} - M_3^{-1} r_{i+2} \\ &\dots \end{aligned}$$

This process can be summarised as

$$x_{i+k} = x_i - M_{\sigma,k}^{-1} r_i$$

where $M_{\sigma,k}^{-1}A$ is inductively determined by

$$\begin{aligned} x_{i+k} &= x_{i+k-1} - M_k^{-1} r_{i+k-1} = x_{i+k-1} - M_k^{-1}(Ax_{i+k-1} - b) \\ &= (I - M_k^{-1}A)x_{i+k-1} + M_k^{-1} b \\ &= (I - M_k^{-1}A)(x_i - M_{\sigma,k-1}^{-1} r_i) + M_k^{-1} b \\ &= x_i - M_k^{-1}(I - AM_{\sigma,k-1}^{-1})r_i + M_{\sigma,k-1}^{-1} r_i \end{aligned}$$

that is,

$$M_{\sigma,k}^{-1} = M_{\sigma,k-1}^{-1} + M_k^{-1}(I - AM_{\sigma,k-1}^{-1}) \tag{132}$$

This also describes the compound application process $x_{i+k} = x_i + t_{i,k}$ with

$$t_{i,1} = M_1^{-1} r_i, \qquad t_{i,k} = t_{i,k-1} + M_k^{-1}(r_i - At_{i,k-1}).$$

Here is $M_{\sigma,3}^{-1}$ explicitly:

$$M_{\sigma,3}^{-1} = \sum_{i=1} M_i^{-1} - \sum_{i_1 > i_2} M_{i_1}^{-1} A M_{i_2}^{-1} + M_3^{-1} A M_2^{-1} A M_1^{-1}.$$

The error propagator is

$$I - M_{\sigma,3}^{-1} A = (I - M_3^{-1}A)(I - M_2^{-1}A)(I - M_1^{-1}A)$$

with obvious extensions to more terms. Proof using equation (130). Here is another equivalent formula:

$$M_{\sigma,3}^{-1} = M_3^{-1}\left[I + N_3 M_2^{-1}[I + N_2 M_1^{-1}]\right] \tag{133}$$

again with obvious extensions to more terms; this is easily seen to be equivalent to (132).

From equation (133) we also get for the case $x_0 = 0$:

$$x_k = M_{\sigma,k}^{-1} f \Rightarrow M_k x_k = f + N_k M_{k-1}^{-1}[\cdots] f. \tag{134}$$

Derive SSOR from forward/backward SOR

Example 1 *Construct the forward and backward SOR splittings*

$$\begin{aligned} A &= D + L + U \\ &= (D+L) - (-U) = M_1 - N_1 \\ &= (D+U) - (-L) = M_2 - N_2 \end{aligned}$$

and combine them to form

$$\begin{aligned} M_{\sigma,2}^{-1} &= M_1^{-1} + M_2^{-1} - M_2^{-1} A M_1^{-1} = M_2^{-1} N_2 M_1^{-1} + M_2^{-1} = M_2^{-1}(N_2 + M \\ &= (D+U)^{-1} D (D+L)^{-1} \end{aligned}$$

Remark 64 *Symmetry and definiteness of compound preconditioners Suppose A and M_2 are symmetric and $M_1 = M_3^t$. Then*

$$A^{-1} - M_{\sigma,3}^{-1} = (I - M_3^{-1}A)(A^{-1} - M_2^{-1})(I - AM_1^{-1}) \tag{135}$$

so the resulting preconditioner is symmetric. This holds for any number of factors. Note that $M_{\sigma,3}$ is symmetric but not necessarily positive definite, even if A and M_2 are. A sufficient condition is (with inequalities in the pd-sense):

$$M_2 \geq A \Rightarrow A^{-1} - M_2^{-1} \geq 0 \Rightarrow A^{-1} - M_{\sigma,3}^{-1} \geq 0 \Rightarrow M_{\sigma,3} \geq A \geq 0.$$

12.4 Convergence of stationary iterative methods

The matrix framework that was used throughout this paper can also be applied to traditional stationary iterative methods and the steepest descent method (see for instance [**?**] and [39]).

Stationary iterative methods in their most basic form update an iterate as

$$x_{n+1} = x_n - M^{-1} r_n \tag{136}$$

where $r_n = Ax_n - f$ and M approximates A. Multiplying by A gives

$$r_{n+1} = r_n - AM^{-1} r_n = (I - AM^{-1}) r_n \tag{137}$$

$$\Rightarrow r_{n+1} = (I - AM^{-1})^n r_1. \tag{138}$$

The error equation for $e_n = x_n - x$ is likewise

$$e_{n+1} = (I - M^{-1}A) e_n.$$

Writing equation (137) on matrix form gives

$$AM^{-1}R = R(I - J),$$

that is, the Hessenberg matrix is of simply bidiagonal form, $I - J$.

Convergence of stationary iterative methods is governed by equation (138). The conclusion is that method converges if and only if $\rho(I - AM^{-1}) < 1$[20]. In practical cases, this is a hard condition to satisfy. If A is an M-matrix coming from a second order PDE, and M is derived by a regular splitting, we find that $\rho(I - AM^{-1}) = 1 - O(h^2)$ (methods such as Jacobi or Gauss-Seidel) or $\rho(I - AM^{-1}) = 1 - O(h)$ (SOR with optimal relaxation parameter), where h is the mesh size.

---

20. Actually, this is convergence *for all initial vectors*. We can have convergence under less limited conditions if the righthand side and initial guess are in the right subspaces. However, because of roundoff this is pretty much irrelevant.

12.5 Parametrisation: Richardson iteration

Parametrising equation (136) gives the Richardson iteration

$$x_{n+1} = x_n - \alpha_n M^{-1} r_n. \tag{139}$$

This updating formula can be written in matrix form as

$$X(I-J) = M^{-1}RD$$

from which we find $R(I-J) = AM^{-1}RD$, and thus

$$AM^{-1}R = R(I-J)D^{-1}. \tag{140}$$

From $r_{n+1} = (I - \alpha_n AM^{-1})r_n$ we get

$$r_{n+1} = \Pi_i^n (I - \alpha_i AM^{-1})r_1 = Q_n(AM^{-1})r_1.$$

If we know the spectrum S of AM^{-1}, we can accelerate convergence by choosing the α_i coefficients to minimize

$$\max_{t \in S} Q_n(t).$$

If AM^{-1} is definite, the optimal polynomial is the Chebyshev polynomial, and this method is known as the Chebyshev semi-iterative method. The above method can also handle indefinite problems.

12.6 Steepest descent

The steepest descent follows from a particular choice of the iteration parameters in the Richardson scheme of equation (139). Of all possible search directions p_i in a general update scheme $x_{i+1} = x_i - \alpha_i p_i$, the choice $p_i = r_i$ gives the steepest descent direction; the value of α_i is chosen to minimise the residual r_{i+1}. Specifically, the optimal parameter in the steepest descent method is

$$\alpha_i = r_i^t r_i / r_i^t A r_i \tag{141}$$

(See section 6.5 for the derivation.)

The line search used in the methods studied earlier in this monograph has the search direction a combination of earlier residuals. Krylov methods such as the Conjugate Gradient algorithm can then be regarded as generalisations of steepest descent where the search direction is in the span of previous residuals, in particular where the combination is such that the residual is orthogonal to all previous search directions; see equation (**??**).

12.7 Recovering CG from Stationary Iteration

After a process of stationary iteration, we have residuals satisfying

$$AM^{-1}R = R(I-J).$$

It is possible to reconstruct from these residuals the ones that would have been generated from a CG process. Both sets of residuals are combinations of the same Krylov sequence; they only differ in that CG residuals are orthogonal. It would then be possible to generate a number of steps of stationary iteration, and take convex combination of these afterwards (see lemma **??**), in such a way as to minimise the residuals in a certain norm.

Thus, we are to find an upper triangular matrix U with column sums $\equiv 1$ which makes for $\bar{R} = RU$ the matrix $\bar{R}M^{-1}\bar{R}$ diagonal. We can then form $\bar{X} = XU$, and these are the solution approximations from CG[21]. The construction of U proceeds by a QR process, during which care is taken that all coefficients in a column sum to one.

This process incurs some storage overhead: in the most naive implementation, we need to retain both the residuals and iterates. Additionally, in order to make $\bar{R}M^{-1}R$ diagonal, we need to have stored both the R and $M^{-1}R$ sequence. However, this is an overestimate of the storage demands in practice. We will now sketch the actual implementation.

In typical stationary iterative methods we solve an equation of the form

$$(D+L)x^{(n+1)} = -Ux^{(n)} + b,$$

so the vectors $r^{(n)}$ and $M^{-1}r^{(n)}$ are never explicitly computed. However, we can derive $M^{-1}r^{(n)} = x^{(n+1)} - x^{(n)}$ at minimal cost, and the coefficients $Lx^{(n)}$ and $-U^{(n)} + b$ are already computed, so assembling $r^{(n)}$ is not very costly. A single relaxation parameter in the stationary iteration can also be easily accomodated, though more complicated relaxations, such as by a diagonal matrix, take more work.

Also, since we are storing the $M^{-1}R$ sequence, we can dispense with storing the iterates except for the first one.

In order to reconstruct MinRes and GMRES from stationary iteration, we use the fact that their residuals are orthogonal under the AM^{-1} inner product, so we store $AM^{-1}R$ instead of $M^{-1}R$. The $AM^{-1}r^{(n)}$ vectors can be derived as $AM^{-1}r^{(n)} = r^{(n+1)} - r^{(n)}$.

Of course, ultimately one wants to reconstruct the iterates of the other sequence, and not just the residuals. If the residuals of one method are convex combinations of those of another method, the iterates follow from the same combination process.

Convex combinations of iterates same as residuals

Lemma 65 *Let R and $\bar{R}$ be residual sequences such that $\bar{R} = RU$ where U has unit column sums, then the iterates sequences X and $\bar{X}$ follow the same relation: $\bar{X} = XU$.*

Proof: We can write in matrix notation $R = AX - fe^t$ where $e^t = (1,\ldots)$. Noting that $e^tU = e^t$, we get

$$\begin{aligned} \bar{R} &= RU \qquad \Leftrightarrow \\ A\bar{X} - fe^t &= (AX - fe^t)U \\ &= AXU - fe^t \end{aligned}$$

which by nonsingularity of A gives the stated result. •

21. We use the term CG generically: our algorithm also works for nonsymmetric matrices, where OrthoMin results; below we will explain how to derive MinRes and GMRES

12.8 Chebyshev semi-iteration

If estimates of the spectrum of the coefficient matrix are known, that is $0 < a < b$ are known such that the spectrum is contained in $[a,b]$, it is possible to construct the iteration polynomials such that the residual is minimized in each iteration.

The theory of this minimization process can be found in section 13.6.

The advantage of the Chebyshev method is that it does not use inner products during the iteration process.

Extension of the Chebyshev method to nonsymmetric systems was discussed by Manteuffel [52]. A procedure for adaptively estimating the spectrum parameters was discussed by Manteuffel [53].

13 Polynomials

Previous section: 12 Next section: 14

In most of this monograph, we do not talk much about polynomials since most of the time any statement regarding them can be expressed in terms of matrices and vectors. This section will gather those results that are better expressed directly in terms of polynomials, or that translate between the polynomial results and the corresponding linear algebra expressions.

This is also where we discuss the convergence speed of conjugacy-based methods; the basic theory of Chebyshev polynomials is in the last section of this chapter (section 13.6).

13.1 Recap of earlier results

Lemma 8:
Let X be a sequence, and let the matrix A and the vector f be given. Define $k_1 = Ax_1 - f$, and let $K = K\langle A, k_1\rangle$. Then the following statements are equivalent.

1. $X = P\langle\{\pi_i\}_{i\geq 1}, A, x_0, f\rangle$
2. There are polynomials $\{\pi_i\}_{i\geq 1}$ such that
$$X(J-I) = (\pi_1(A)k_1, \pi_2(A)k_1, \ldots).$$
3. There are polynomials $\{\pi_i\}_{i\geq 1}$ such that
$$X(J-E_1) = (\pi_1(A)k_1, \pi_2(A)k_1, \ldots).$$
4. There is an upper triangular matrix U such that
$$X(J-I) = KU.$$
5. There is an upper triangular matrix U such that
$$X(J-E_1) = KU.$$

Lemma 9:
Let a matrix A a vector f and a sequence X be given. Then

$$\begin{aligned} & \exists_{\{\pi_i \in \mathbf{P}^{(\mathbf{n})}\}} : X = P\langle \{\pi_i\}_{i\geq 1}, A, x_0, f\rangle \\ \Leftrightarrow \quad & \exists_{\{\phi_i \in \mathbf{P}^{(\mathbf{n},\mathbf{1})}\}} : R\langle A, X, f\rangle = [\phi_i(A) r_1] \end{aligned}$$

and the ϕ_i and π_i polynomials are related by

$$\phi_{i+1}(x) = 1 + x\pi_i(x).$$

Theorem 32, item 7:
There are a nonsingular matrix M and polynomials $\{\pi_i\}_{i\geq 1}$ such that

$$r_i = \pi_i(AM^{-1})r_1, \qquad \deg(\pi_i) = i-1, \qquad \pi_i(0) = 1.$$

13.2 Orthogonal polynomials

Some standard theory of orthogonal polynomials [**?**].

Consider polynomials generated by a recurrence

$$\phi_{n+1} = \alpha_n t\phi_n - \sum^{n} \beta_i \phi_i \tag{142}$$

with an inner product $(\phi_i, \phi_j) = \int \phi_i(t)\phi_j(t)dt$ or $(\phi_i, \phi_j) = \sum_k \phi_i(x_k)\phi_j(x_k)$.

Orthogonal polynomials satisfy three-term recurrence

Lemma 66 *Let $\{\phi_i\}$ be a sequence of orthogonal polynomials, that is, $(\phi_i, \phi_j) = 0$ for $i \neq j$. Then the polynomials satisfy a three-term recurrence*

$$\phi_{n+1}(t) = (\alpha_n t - \beta_n)\phi_n(t) - \gamma_n \phi_{n-1}(t), \qquad \phi_{-1} \equiv 0, \phi_0 \equiv A_0. \tag{143}$$

Proof: Let

$$\phi_{n+1} = \alpha t\phi_n - \sum^{n} \beta_i \phi_i.$$

From orthogonality, $(\phi_{n+1}, \phi_k) = 0$ for $k \leq n$ iff

$$\alpha(\phi_k, t\phi_n) = \sum_i \beta_i(\phi_k, \phi_i) = \beta_k \|\phi_k\|.$$

This determines the coefficients β_k. If $k < n-1$, $(\phi_k, t\phi_n) = (t\phi_k, \phi_n) = 0$, since $t\phi_k$ has degree $< n$ and ϕ_n is orthogonal to all polynomials of lesser degree. •

This lemma for orthogonal polynomials has an exact counterpart for residual sequences.

Residual sequences generated by symmetric matrix satisfy three-term recurrence

Lemma 67 *Let r_i be a sequence generated from*

$$r_{n+1} = \alpha^{(n)} A r_n - \sum_i^n \beta_i^{(n)} r_i,$$

where A is symmetric. Then the vectors satisfy a three-term recurrence

$$r_{n+1} = (\alpha^{(n)} A - \beta_n^{(n)}) r_n - \beta_{n-1}^{(n)} r_{n-1}.$$

Proof: Analogous to the above proof we determine the β coefficients as

$$\beta_k/\alpha = r_k^t A r_n / \|r_k\|^2.$$

We then observe that if A is symmetric and $k < n-1$,

$$A^t r_k = A r_k \in [r_1, \ldots, r_{k+1}],$$

so $r_k^t A r_n = 0$ for $k+1 < n$. •

This proof is essentially that of lemma 47: the full recurrence is expressed as $AR = RH$, from which by orthogonality of the r_i vectors

$$h_{kn} = \frac{r_k^t A r_n}{\|r_k\|^2}.$$

If A is symmetric, for $k < n-1$

$$h_{kn} = \frac{\|r_n\|^2}{\|r_k\|^2} h_{nk} = 0.$$

Question: *Where does symmetry come into the polynomials proof?*

13.3 CG polynomials

The update relations for CG are

$$\begin{aligned} M^{-1} r_{i+1} &= M^{-1} r_i - M^{-1} A p_i \alpha_i \\ p_i &= M^{-1} r_i - p_{i-1} \beta_i \end{aligned}$$

Since all $\alpha_i > 0$,

$$M^{-1} r_{i+1} = \rho_{i+1}(M^{-1}A) M^{-1} r_1$$

where

$$\rho_{i+1}(x) = (-1)^i |c_{i+1}^{(i+1)}| + \cdots$$

That is, the coefficients of the highest exponent are of alternating sign. Contrast this with the Arnoldi procedure, where

$$z_{i+1} = c_{i+1}(A z_i - c_i z_i - \cdots$$

13.4 Polynomials of search directions

We gather some results on the polynomials associated with residuals and search directions.

Residual polynomials satisfy recurrence with coefficients in H

Lemma 68 *Let R be a residual sequence satisfying $AR = RH$, and let polynomials ρ_i be such that $r_i = \rho_i(A)r_1$. Then:*

$$h_{n+1n}\rho_{n+1}(t) = (t - h_{nn})\rho_n(t) - \sum_{k=1}^{n-1} h_{kn}\rho_k(t)$$

We present the following results in the context of Lanczos orthogonalization. For Arnoldi orthogonalization, the left sequence can simply be ignored.

Polynomials for residuals and search directions

Lemma 69 *Let residuals r_i and search directions p_i, and corresponding left sequences $\tilde{r}_i$ and $\tilde{p}_i$, be defined by coupled two-term recurrences (theorem 32, item 5):*

$$r_{i+1} = r_i - Ap_i\alpha_i^{(r)}, \qquad p_i = M^{-1}r_i - p_{i-1}\beta_i^{(r)}$$

and

$$\tilde{r}_{i+1} = \tilde{r}_i - A^t\tilde{p}_i\alpha_i^{(\ell)}, \qquad \tilde{p}_i = M^{-t}\tilde{r}_i - \tilde{p}_{i-1}\beta_i^{(\ell)}.$$

There are polynomials ρ_i, $\tilde{\rho}_i$, π_i, and $\tilde{\pi}_i$ of degree $i-1$, such that

$$\begin{cases} r_i = \rho_i(AM^{-1})r_1, & p_i = M^{-1}\pi_i(AM^{-1})r_1 \\ \tilde{r}_i = \tilde{\rho}_i(A^tM^{-t})\tilde{r}_1, & \tilde{p}_i = M^{-t}\pi_i(A^tM^{-t})\tilde{r}_1 \end{cases} \tag{144}$$

The polynomials satisfy recurrences

$$\begin{cases} \rho_{i+1}(x) = \rho_i(x) - x\pi_i(x)\alpha_i^{(r)} \\ \tilde{\rho}_{i+1}(x) = \tilde{\rho}_i(x) - x\tilde{\pi}_i(x)\alpha_i^{(\ell)} \\ \pi_{i+1}(x) = \rho_{i+1}(x) - \pi_i(x)\beta_i^{(r)} \\ \tilde{\pi}_{i+1}(x) = \tilde{\rho}_{i+1}(x) - \tilde{\pi}_i(x)\beta_i^{(\ell)} \end{cases} \tag{145}$$

Proof: This follows immediately from

$$r_{i+1} = \rho_i(AM^{-1})r_1 - AM^{-1}\pi_i(AM^{-1})\alpha_i^{(r)}r_1$$

and

$$p_{i+1} = M^{-1}\rho_{i+1}(AM^{-1})r_1 - M^{-1}\pi_{i-1}(AM^{-1})\beta_i^{(r)}r_1,$$

(and similar for the left sequence) from which

$$\begin{cases} \rho_{i+1}(AM^{-1}) = \rho_i(AM^{-1}) + AM^{-1}\pi_i(AM^{-1})\alpha_i^{(r)}; \\ \tilde{\rho}_{i+1}(A^tM^{-t}) = \rho_i(A^tM^{-t}) + A^tM^{-t}\pi_i(A^tM^{-t})\alpha_i^{(\ell)} \\ M^{-1}\pi_{i+1}(AM^{-1}) = M^{-1}\rho_{i+1}(AM^{-1}) + M^{-1}\pi_i(AM^{-1})\beta_i^{(r)} \\ M^{-t}\tilde{\pi}_{i+1}(A^tM^{-t}) = M^{-t}\tilde{\rho}_{i+1}(A^tM^{-t}) + M^{-t}\tilde{\pi}_i(A^tM^{-t})\beta_i^{(\ell)} \end{cases}$$

•

13.5 Constructing a sequence from a polynomial

Consider the scheme

$$\begin{aligned} r_i &= Ax_i - b \\ x_{i+1} &= x_i - \alpha_i r_i \end{aligned}$$

The second line gives $r_{i+1} = (I - \alpha_i A) r_i$. Continuing,

$$r_{i+1} = P_i(A) r_1, \qquad P_i(x) = \Pi(1 - \alpha_i x).$$

This means that if we have a suitable polynomial and its zeros $-\alpha_i^{-1}$, we can apply it to construct a polynomial sequence. This is the basic idea behind Richardson iteration; section 12.5.

This could be a cool idea if more than one system with the same coefficient matrix is to be solved. The coefficients α_i can be computed from the first system at some expenditure (for instance with a GMRES method), after which the other sequences can be solved at less expense.

13.6 Chebyshev polynomials

From the trigonometric identity

$$\cos[(k+1)\theta] = 2\cos(\theta)\cos(k\theta) - \cos[(k-1)\theta] \tag{146}$$

we find that $\cos(k\theta)$ can be expressed as a polynomial of degree k in $\cos(\theta)$. Proof by induction, where the cases $k = 0, 1$ are trivial.

Denoting these polynomials as

$$T_k(\cos\theta) = \cos(k\theta) \tag{147}$$

we get the Chebyshev polynomials. They satisfy the recurrence

$$\begin{aligned} &T_0(x) = 1, \quad T_1(x) = x, \\ &T_{k+1}(x) = 2T_k(x) - T_{k-1}(x) \end{aligned}$$

Obviously, we can extend the domain of definition beyond the interval $[-1, 1]$.

Since the function $\cosh(k\theta)$ also satisfies the trigonometric recurrence equation (146), we find that

$$T_k(\cosh(\theta) = \cosh(k\theta)$$

and

$$T_k(x) = \begin{cases} \cos[k\cos^{-1}(x)] & \text{for } |x| \leq 1 \\ \cosh[k\cosh^{-1}(x)] & \text{for } x \geq 1 \\ (-1)^k \cosh[k\cosh^{-1}(-x)] & \text{for } x \leq -1 \end{cases}$$

or

$$T_k(x) = \frac{1}{2}[(x + \sqrt{x^2-1})^k + (x - \sqrt{x^2-1})^k].$$

From equation (147) obviously

$$\begin{cases} \max_{|x|\leq 1} |T_k(x)| = 1 \\ T_k(x_i) = (-1)^i & \text{for } x_i = \cos(i\pi/k). \end{cases}$$

Outside of the $[-1,1]$ interval, the polynomials grow quickly. In a way, they grow quicker than any other polynomial. This is the basis for the standard Chebyshev minimization theorem.

Let $0 < a < b$ and consider the mapping $x \mapsto (b+a-2x)/(b-a)$; this maps

$$\begin{Bmatrix} b \\ a \\ 0 \end{Bmatrix} \rightarrow \begin{Bmatrix} -1 \\ 1 \\ \frac{b+a}{b-a} > 0 \end{Bmatrix}$$

Consider the shifted and scaled Chebyshev polynomial

$$\tilde{T}_k(x) = T_k\left(\frac{b+a-2x}{b-a}\right) / T\left(\frac{b+a}{b-a}\right);$$

the fact that T_k grows faster than any other polynomial is expressed by the fact that $\tilde{T}_k$ is less than any other polynomial on the interval $[a,b]$ among those polynomials that satisfy $P(0) = 1$.

Theorem 70 *Let $0 < a < b$, let $\mathbf{P}^{(\mathbf{n},\mathbf{1})}$ be the set of polynomials of degree n with $P(0) = 1$, and let $\tilde{T}_n(x) = T_n((b+a-2x)/(b-a))/T((b+a)/b-a))$ be the scaled and shifted Chebyshev polynomial, then*

$$\max_{x\in[a,b]} |\tilde{T}_n(x)| = \min_{P\in\mathbf{P}^{(\mathbf{n})}} \max_{x\in[a,b]} |P(x)|$$

Proof: We will use the following properties of $\tilde{T}_n$:

$$\tilde{T}_n(0) = 1, \qquad \begin{cases} a = x_0 < \cdots < x_n = b \\ \forall_i \colon |\tilde{T}_n(x_i)| = M = \max_{x\in[a,b]} |\tilde{T}_n(x)| \end{cases}$$

(Just to stress the relevant bit, it is essential that the maximum value on $[a,b]$ is assumed in $n+1$ points.)

Now assume that $P \in \mathbf{P}^{(\mathbf{n},\mathbf{1})}$ is such that

$$\max_{x\in[a,b]} |P(x)| < M,$$

then in particular

$$\forall_i \colon |P(x_i)| < M.$$

This implies that with $r = \tilde{T}_n - P$:

$$\begin{aligned} P(x_i) > 0 \quad &\Rightarrow \quad r(x_i) > 0 \\ P(x_i) < 0 \quad &\Rightarrow \quad r(x_i) < 0 \end{aligned}$$

Since r alternates on the $k+1$ points x_i, there must be k zeros y_i:

$$\forall_{i=0\ldots k-1} \colon y_i \in (x_i, x_{i+1}) \text{ and } r(y_i) = 0.$$

However, $r(0) = 0$ too, so r is a polynomial of degree $\leq k$ with $k+1$ zeros. By contradiction, no polynomial P with the stated properties can exist. •

14 Convergence theory

Previous section: 13 Next section: 15

Combining lemma 9, which effectively says that all possible choices for the i-th residual are in the same subspace, and corrolary 41 which states that the orthogonal residual in minimal in that subspace, we get the statement that

$$\|r_n\|_{A^{-1}} = \min_{\pi_n \in \mathbf{P}^{(n)}} \|\pi_n(AM^{-1})r_1\|.$$

For any given polynomial, we can bound the residual size by a bound on the spectrum.

Convergence is dominated by estimate of polynomial on spectrum

Lemma 71 *Let S be a set that contains the spectrum of AM^{-1}, let $\{\pi_i\}$ be the polynomials of some polynomial iterative method, and let M_i be such that*

$$\lambda \in S\colon |\pi_i(\lambda)| \leq M_i,$$

then

$$\|r_n\| \leq M\|r_1\|$$

Proof: This follows from $r_n = \pi_n(AM^{-1})r_1$. •

This leads to the basic theorem of CG convergence.

Convergence of CG determined by best polynomial bound on spectrum

Theorem 72 *Let R be the residuals of the CG method, let S contain the spectrum of AM^{-1}, and define a polynomial-specific bound M_π such that*

$$\forall_{\lambda \in S}\colon |\pi(\lambda)| \leq M_\pi$$

then

$$\|r_n\|_{A^{-1}} = \min_{\pi_n} M_{\pi_n} \|r_1\|_{A^{-1}}.$$

This means that we can bound the residual size by bounding various polynomials on (a set containing) the spectrum.

14.1 Speed of convergence of Conjugate Gradients

For the symmetric conjugate gradients method a convergence bound is easy to derive. From theorems 72 and 70 it follows that we need to bound

$$T_n\left(\frac{b+a-2x}{b-a}\right) \mid / \left(\frac{b+a}{b-a}\right)$$

with $a = \lambda_{\min}$, $b = \lambda_{\max}$. The denominator is bound by 1, so we can ask what the smallest n is for which $T_n(b+a/b-a) > 1/\varepsilon$. Let $\kappa = \lambda_{\max}/\lambda_{\min}$ be the spectral condition number, then

$$T_n\left(\frac{b+a}{b-a}\right) = T_n\left(\frac{\kappa+1}{\kappa-1}\right) = \frac{1}{2}\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\right] \geq \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n$$

Use the fact that

$$\ln[(\sqrt{\kappa}+1)/(\sqrt{\kappa}-1)] > 2/\sqrt{\kappa}$$

to see that

$$n > (1/2)\sqrt{\kappa}\ln(2/\varepsilon) \Rightarrow T_n(b+a/b-a) > 1/\varepsilon.$$

14.2 Error reduction with isolated eigenvalues

Most convergence statements estimate speed of convergence in terms of the size of some eigenvalue cluster. Outlying eigenvalues are generally considered to delay convergence. Results to this effect were proved by Campbell et al [11]:

- If there is a single eigenvalue cluster with radius ρ, and d is the sum of the degrees of the minimal polynomials of the outliers, then
$$\|r_{d+k}\| \leq C(\rho)\rho^k \|r_0\|.$$
- In the case of P cluster, a similar estimate
$$\|r_{d+kP}\| \leq C(\rho)(\sigma^{P-1}\rho)^k \|r_0\|$$
holds, where σ is the maximal distance between clusters.

14.3 Stopping tests

Many people use the test $\|r_i\| \leq \varepsilon \|r_1\|$. This is not a good idea [7, 45, 3].

14.3.1 Backward error analysis

Ultimately, in error analysis we want to bound the distance between the computed and the true solution. For instance, if
$$\frac{\|x-\hat{x}\|}{\|x\|} < 10^{-n}$$
then the computed solution has n correct digits. Of course, this forward error can not be computed.

However, we can bound it if we take a small detour.

Backward error analysis considers that the computed solution is the exact solution of a perturbed system
$$(A+\delta A)x = b+\delta b. \tag{148}$$
Clearly, we want the relative perturbations $\|\delta A\|/\|A\|$, $\|\delta b\|/\|b\|$ to be small, ideally machine precision, because that would correspond to any normal contamination of the representation of a continuous problem as a matrix equation.

The normwise backward error is then defined as the minimum ε such that
$$\max\{\frac{\|\delta A\|}{\|A\|}, \frac{\|\delta b\|}{\|b\|} \leq \varepsilon \tag{149}$$
where δA, δb satisfy equation (148).

The following theorem [66] has a more practical form of the backward error.

Backward error can be computed from residual and solution

Theorem 73 *The normwise backward error defined in equation (149) satisfies*

$$\varepsilon = \frac{\|r\|}{\|A\|\|x\| + \|b\|} \tag{150}$$

Using the backward error, we can bound the forward error. Consider the perturbed solution of a perturbed system:

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

Our goal is to bound $\|\delta x\|/\|x\|$, assuming that we satisfy a backward error bound as in equation (149).

Using $Ax = b$ we easily get

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

Taking norms gives

$$\|\delta x\|(1 - \|A^{-1}\|\|\delta A\|) \leq \|A^{-1}\|(\|\delta b\| + \|\delta A\|\|x\|)$$

which, using $\|b\| \leq \|A\|\|x\|$ becomes

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

Introducing $\kappa = \|A\|\|A^{-1}\|$, this becomes

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\kappa}{1 - \kappa\frac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right) \leq \frac{2\kappa\varepsilon}{1-\kappa\varepsilon} \tag{151}$$

where we assume the backward error bound to hold, and we assume that $\kappa\varepsilon < 1$.

The theory of Oettli and Prager [57] that for with $r = Ax - b$ and

$$\varepsilon = \max_i \frac{|r_i|}{(E|x| + b)_i}$$

we have that

$$(A+\delta A)x = b+\delta b, \qquad |\delta A| \leq \varepsilon E \text{ and } |\delta b| \leq \varepsilon b$$

This means that by minimizing ε we obtain the solution to a nearby system.

Arioli et al. [1] proposed a variant on this:

$$\varepsilon = \frac{\|r\|}{\|A\|\|x\| + \|b\|}$$

(Check this!)

14.3.2 Eigenvalue based estimates

Ultimately, we want

$$\|x - x_i\|/\|x\| \leq \varepsilon.$$

For this it is enough to have

$$\|x - x_i\|/\|x_i\| \leq \varepsilon/(1+\varepsilon),$$

and, since $x - x_i = A^{-1}(b - Ax_i) = -A^{-1}r_i$, we get for symmetric matrices the stopping criterion

$$\|r_i\| \leq \mu_1\|x_i\|\varepsilon/(1+\varepsilon),$$

where μ_1 is the smallest eigenvalue. Kaasschieter [45] proposes an efficient way of computing this smallest eigenvalue from the CG polynomials by using bisection with the previous estimate as starting guess.

14.4 Breakdown conditions

In this section we investigate the breakdown of iterative methods, that is, conditions under which an inner product appearing in the nominator of a scalar expression is zero.

14.4.1 General theory

In section 8.2.8 we considered conjugacy-based methods under a general inner product that makes the residuals satisfy

$$R^t N^{-1} R \text{ upper triangular.}$$

We did not consider whether, given an inner product N^{-1}, this construction is well-defined a priori. For this, recall that for polynomial iterative methods,

$$R = KU, \qquad \text{with } U \in \mathbf{U}^{(1)} \text{ and } K = K\langle A, x_0, f\rangle AM^{-1}, r_1.$$

Constructing R is then equivalent to constructing U. In the next lemma we will show that U can be constructed inductively, as long as K is independent under N^{-1}.

Arnoldi orthogonal R constructable from independent K

Lemma 74 *Let $K = K\langle AM^{-1}, r_1\rangle$ and N a symmetric (general) nonsingular matrix It is possible to construct U_{n+1} such that $R_{n+1}^t N^{-1} R_{n+1}$ is diagonal (upper triangular), where the sequence R is constructed from $R = KU$, iff the first $n+1$ columns of K are independent.*

Proof: Let R_n, U_n be the initial n columns of R, U. Suppose inductively that $R_n^t N^{-1} R_n$ is diagonal. In order to let r_{n+1} be N^{-1}-orthogonal to R_n, we need to solve the $n+1$-st column, u_{n+1}, of U from the overdetermined system

$$U_n^t \bar{N} u_{n+1} = \left.\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}\right\} n$$

where $\bar{N}$ is the $n \times n+1$ primary subblock of $K_n^t N^{-t} K_{n+1}$. This determines u_{n+1} up to scaling. We scale it so that $u_{1,n+1} = 1$, so the system to be solved now becomes

$$\bar{N} \begin{pmatrix} 0 \\ u_{2n+1} \\ \vdots \\ u_{n+1\,n+1} \end{pmatrix} = \bar{N} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Now it follows that $R_n^t N r_{n+1} = 0$, and for symmetric N, $R_{n+1}^t N R_{n+1}$ is diagonal.

The reverse assertion is trivial: if K has dependent columns, clearly no orthogonal sequence can be constructed from it. •

The requirement for the Krylov sequence to be independent shows up in a different guise in section 16.11.1.

For the case of Lanczos orthogonalisation, we have an added condition on the existence of the residual sequence.

Lanczos orthogonal R exists if K independent and no accidental $s_i^t r_i = 0$

Lemma 75 *Show this*

The Lanczos breakdown condition can be prevented with a look-ahead strategy.

14.4.2 Iterative forms of the breakdown condition

In remark 79 we observed that breakdown is associated with the Hessenberg matrix becoming reducible, that is, with $h_{n+1n} = 0$ for some n. In the search-direction formulation of Krylov methods, this corresponds to

$$h_{n+1n} = -d_n^{-1} = -\frac{p_n^t A p_n}{r_n^t M^{-1} r_n}.$$

15 Eigenvalues

Previous section: 14 Next section: 16

Arnoldi methods perform a reduction $V^t AV = H$ with H of upper Hessenberg form (in the symmetric case $H = T$ of tridiagonal form), and $V^t V = I$, so this is a similarity transformation, and the spectrum of H is that of A.

The Lanczos method uses different sequences V and W with $W^t AV = T$ of tridiagonal form and $W^t V = I$. Observing that $W^t VW^t = W^t$, when also $VW^t = I$, we can transform an eigenequation

$$Au = \mu u \Rightarrow W^t AVW^t u = \mu W^t u \Rightarrow TW^t u = \mu W^t u$$

and again we see that the spectrum of T is that of A.

15.1 Ritz values and Harmonic Ritz values

In the finite case, let V_n be the block containing the first n Arnoldi vectors, and $A_n = V_n^t AV_n$, then the eigenvalues and eigenvectors of A_n are called the Ritz values and Ritz vectors.

With $W = AV$ and using an Arnoldi method that makes $W^t W = I$, we get

$$W^t A^{-1} Wy = V^t AVy = \lambda^{-1} V^t A^t AVy = \lambda^{-1} W^t Wy = \lambda^{-1} y.$$

In other words, making AV orthonormal makes that the eigenvalues of $V_n^t AV$ approximate λ^{-1} where λ is an eigenvalue of A. The λ values thus obtained are called harmonic Ritz values [62, 55, 33].

Question: *Here is another definition: for a set S, x is a Ritz vector with corresponding Ritz value* θ *if*

$$x^t(Ax - \theta x) = x \qquad \forall_{x \in S}.$$

Is this the same?

15.2 Eigenvalues of the Hessenberg matrix

In the following, read "AM^{-1}" any time "A" is used.

15.2.1 Relationship between eigenvalues of A and H

After n steps of an Arnoldi process we have an equation $AV_n = V_{n+1}H_{[n+1,n]}$ or

$$AV_n = V_nH_n + v_{n+1}[0,\ldots,h_{n+1,n}].$$

Let $H_nx = \lambda x$ be an eigen equation and let $y = V_nx$, then

$$\begin{aligned} Ay &= AV_nx = V_nH_nx + v_{n+1}[0,\ldots,h_{n+1,n}]x \\ &= \lambda y + v_{n+1}h_{n+1,n}x_n \end{aligned}$$

With $r_x = Ay - \lambda y$, the size of this residual is

$$|r_x| = \|v_{n+1}\|\,|h_{n+1,n}|\,|x_n|.$$

If we normalise x, so that $x_n \leq 1$, and we orthonormalise the v_i vectors, we see that $\|r_x\| \leq |h_{n+1n}|$, so we can monitor the convergence by the subdiagonal elements of the Hessenberg matrix.

16 Other aspects of the iterative solution process

Previous section: 15 Next section: 17

16.1 Inventory of all upper triangular matrices

Let a preconditioned polynomial iterative method $X = P^{\pi}\langle\{\pi_i\}_{i\geq 1}, A, M, x_0, r_0\rangle$, its residual sequence $R = R\langle A, X, f\rangle$, and the Krylov sequence of the first residual $K = K\langle A, x_0, f\rangle$ be given. Let $U \in \mathbf{U}^{(\mathbf{1})}$ be such that

$$R = KU,$$

then lemma 17

$$X(J-E) = KU_2 \qquad \text{with } U_2 = U_{[2:*,2:*]}.$$

Let H be the upper Hessenberg matrix (with zero column sums by lemma 23)

$$H = U^{-1}JU \qquad \text{then} \qquad AR = M^{-1}RH.$$

Search directions P are defined as $PU_3 = M^{-1}R$ where

$$H = (I-J)D^{-1}U_3^{-1}.$$

16.2 The symmetrisable case

For further reading: the material in this section is not referred to elsewhere.

In the foregoing we have repeatedly seen how nonsymmetric systems need different methods than symmetric ones. There is a simple exception, namely for the case where A might be nonsymmetric, but $\hat{A} = G^{1/2}AG^{-1/2}$ is symmetric for a symmetric matrix $G^{1/2}$. In that case we draw up the equations for an unpreconditioned conjugate gradient method with matrix $\hat{A}$:

$$\hat{A}PD = R(I-J), \qquad P(I-U) = R, \qquad r_1 = \hat{A}x_1 - G^{1/2}f \tag{152}$$

which defines a polynomial method X for $\hat{A}^{-1}G^{1/2}f = G^{1/2}A^{-1}f$. The sequence $\bar{X} = G^{-1/2}X$ is then a method for $A^{-1}f$.

Since $R = G^{1/2}(A\hat{X} - f)$, we will be interested in transformed sequences

$$\tilde{R} = G^{-1/2}R, \qquad \tilde{P} = G^{-1/2}P. \tag{153}$$

In terms of these we get

$$A\tilde{P}D = \tilde{R}(I-J), \qquad \tilde{P}(I-U) = \tilde{R}, \tag{154}$$

which we use as a basis for computation in terms of A.

It remains to compute the quantities

$$d_{ii} = r_i^t r_i / p_i^t \hat{A} p_i, \qquad u_{ii+1} = r_{i+1}^t r_{i+1} / r_i^t r_i,$$

for which we use

$$p_i^t \hat{A} p_i = \tilde{p}_i^t GA\tilde{p} \qquad \text{and} \qquad r_i^t r_i = \tilde{r}_i G\tilde{r}.$$

In these last identities we use the symmetry of $G^{1/2}$.

To update the solution, we note (see above) that we are really interested in $G^{-1/2}X$, and the update for this we can easily read from that for $\tilde{R} = G^{-1/2}R$.

The method now becomes:

Algorithm: Symmetrised Conjugate Gradient Method

Algorithm 9 *Let x_1 be given, and compute $r_1 = Ax_1 - f$. Then iterate:*

- *Compute*
$$\rho_i = r_i^t G r_i.$$
- *For $i > 1$, update*
$$p_i = r_i + \rho_i / \rho_{i-1} p_{i-1}.$$
- *Compute*
$$\pi_i = p_i^t GA p_i.$$
- *Update*
$$r_{i+1} = r_i - Ap_i(\rho_i/\pi_i) \qquad \text{and} \qquad x_{i+1} = x_i - p_i(\rho_i/\pi_i).$$

We note that left and right preconditioned systems (see section 16.14) fall under the symmetrisable case: if A and M are symmetric, $M^{-1}A$ and AM^{-1} are not, but can be symmetrised as described above. Algorithm 9 is then essentially the same as the CG-variant described in section 8.2.6.

Making instead of equation (153) the substitution

$$\tilde{R} = G^{-1/2}R, \qquad \tilde{P} = G^{1/2}P, \tag{155}$$

we get from equation (152)

$$A\tilde{P}D = \tilde{R}(I-J), \qquad \tilde{P}(I-U) = G\tilde{R},$$

which are the ordinary generating equations for CG. The coefficients are computed from

$$r_i^t r_i = \tilde{r}_i^t G\tilde{r}_i, \qquad p_i^t \hat{A} p_i = \tilde{p}_i^t AG\tilde{p}_i.$$

Applying the above to the nonsymmetric system AM^{-1}, with both A and M symmetric, and the choice $G = M^{-1}$, we see that we have rederived the preconditioned conjugate gradients method.

Question: *Why do we have p^tAGp instead of p^tAp?*

16.3 Iterative solution as solving reduced systems

For further reading: the material in this section is not referred to elsewhere.

One of the characterisations of polynomial iterative methods, theorem 32[item 4], was $X(I-J) = M^{-1}RU$ for some upper triangular U. By lemma 5, this is equivalent to

$$X(E_1 - J) = M^{-1}RV$$

for another upper triangular matrix V. From this we get $R(E_1 - J) = AM^{-1}RV$, and if R is M^{-1}-orthogonal we find $R^tM^{-1}AM^{-1}RV = R^tM^{-1}R(E_1 - J)$, or, in the nth column:

$$\begin{pmatrix} \|r_1\|^2_{M^{-1}} \\ 0 \\ \vdots \\ 0 \\ -\|r_{n+1}\|^2_{M^{-1}} \end{pmatrix} = R^t_{n+1}M^{-1}AM^{-1}R_nv_n,$$

which we restrict[22] to

$$\begin{pmatrix} \|r_1\|^2_{M^{-1}} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = R^t_nM^{-1}AM^{-1}R_nv_n.$$

The generating formula for x_i now becomes

$$x_{i+1} - x_1 = \|r_1\|^2R_n(R^t_nM^{-1}AM^{-1}R_n)^{-1}e_1.$$

We see that the columns of V follow from the solution of a reduced system that is derived from the previous by the addition of a row and column. From lemma 17 we know that if $x_{n+1} - x_1 = M^{-1}R_nv_n$, then $r_n =$.

22. The fact that $r^t_{n+1}M^{-1}AM^{-1}R_nv_n = -\|r_{n+1}\|^2_{M^{-1}}$ is probably of no use.

From $X(I-J) = M^{-1}RU$ we get similarly

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ \|r_n\|^2_{M^{-1}} \end{pmatrix} = R_n^t M^{-1} A M^{-1} R_n v_n,$$

so

$$x_{i+1} - x_i = \|r_n\|^2 R_n (R_n^t M^{-1} A M^{-1} R_n)^{-1} e_n.$$

16.4 $AV_n = V_n H + something$

By talking about whole sequences we have neatly avoided certain complications, that would arise from considering only an initial part, which are not essential to the gist of the story. Here we flesh out one of those points, only to show that indeed it does not matter.

We have many times given the equations $AK = KJ$ and $AR = RH$ for a Krylov sequence and combinations $R = KU$ thereof. In this, $H = U^{-1}JU$. To make a statement about initial parts, for instance K_n, of these sequences, we write

$$AK_n = K_{n+1}J_{[n+1,n]} \tag{156}$$

where $J_{[n+1,n]}$ denotes the $n+1 \times n$ initial part of J. In order to get K_{n+1} in the left hand side, we have to add to the right hand side:

$$AK_{n+1} = K_{n+1}J_{n+1} + Ak_{n+1}e_{n+1}^t. \tag{157}$$

The finite form of $R = KU$ is $R_{n+1} = K_{n+1}U_{n+1}$, giving

$$AV_{n+1} = V_{n+1}U_{n+1}^{-1}J_{n+1}U_{n+1} + Ak_{n+1}e_{n+1}^t u_{n+1n+1},$$

that is,

$$AV_{n+1} = V_{n+1}H_{n+1} + Ak_{n+1}e_{n+1}^t u_{n+1n+1},$$

with $H_{n+1} = U_{n+1}^{-1}J_{n+1}U_{n+1}$. Reducing this again to n columns gives

$$AV_n = V_{n+1}H_{[n+1,n]}$$

with

$$H_{[n+1,n]} = U_{n+1}^{-1}J_{n+1}U_{[n+1,n]} = U_{n+1}^{-1}J_{[n+1,n]}U_n$$

which is what we found from $AV = VH$ in the first place.

Instead of as equation (157), equation (156) could also have been split as

$$AK_n = K_nJ_n + k_{n+1}e_n^t,$$

which is often with less precision rendered as

$$AK_n = K_nJ_n + re_n^t \qquad \text{with } K_n^t r = 0.$$

16.5 Singular matrices

Let A be singular, and let N span the nullspace: $AN = 0$. This is also called the right nullspace; the left nullspace is the nullspace of A^t. The left and right nullspace need not be the same: consider

$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \bar{0}; \qquad \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} = \bar{0}^t$$

Let N_ℓ be the left nullspace $N_\ell^t A = 0$, then in order for $Ax = b$ to have a solution, we need consistency: $N_\ell^t b = 0$.

Solutions to $Ax = b$ are not unique. With N the nullspace $AN = 0$ and $x' = x + Ny$, we also have $Ax' = b$. Uniqueness is guaranteed by requiring $N^t x = 0$ for the solution.

Since $x_{n+1} = x_0 + \sum_{k \leq n} \alpha_k (M^{-1}A)^k r_0$, it is enough for uniqueness if the range of M^{-1} does not have components in the nullspace of A, that is, $N^t M^{-1} = 0$. We can do this by post-processing M^{-1}:

$$\text{let } M'^{-1} = (I - N(N^tN)^{-1}N^t)M^{-1}, \text{ then } N^t M'^{-1} = 0.$$

Question: *If M is symmetric, is M' too, in some sense?*

16.6 Iterating in subspaces

For further reading: the material in this section is not referred to elsewhere.

16.6.1 approach 1

This section is based on [56].

Let E be a basis for the left-nullspace of A, that is, $E^t A = 0$. For any vector x, $E^t Ax = 0$, that is, Ax is orthogonal to the nullspace. Now let $E^t r_i = 0$ for $i \leq n$, then from

$$r_{n+1} = \alpha AM^{-1} r_n + \sum_{i \leq n} \beta_i r_i$$

we find that $E^t r_{n+1} = 0$, that is, given an initial residual that is orthogonal to the nullspace, each subsequent one will be too. This statement is independent of the coefficients chosen; in particular, it does not use orthogonality of any kind. Also, note that M^{-1} appears as a right multiplicator, hence we can talk about the nullspace of A without involving M in the story.

We need to force the first residual to be orthogonal to E. Let x be arbitrary; we want to calculate f such that with $x_1 = x - f$ the vector r_1 is E-orthogonal. Consider the identities

$$\begin{aligned} E^t r_1 &= E^t(Ax_1 - b) \\ &= E^t(Ax - Af - b) \\ &= E^t r - E^t A f \end{aligned}$$

Now let d be such that $f = Ed$, then solving

$$E^t AEd = E^t r$$

will give d, and therefore f.

We can construct a singular matrix as follows:

$$\tilde{A} = A(I - E(E^tAE)^{-1}E^tA).$$

Clearly $E^t\tilde{A} = 0$.

16.6.2 approach 2

Here's another way of looking at deflated system solving. Let E describe a low-dimensional subspace. Defining

$$P = I - E(E^tE)^{-1}E^t, \qquad Q = I - P = E(E^tE)^{-1}E^t$$

gives

$$E^tQ = E^t, \quad E^tP = 0, \quad QE = E, \quad PE = 0.$$

We now try to find a solution of the form $x+y$ such that $A(x+y) = b$. We try to find x,y such that

$$PAx = Pb, \quad PAy = 0, \quad QAy = Qb, \quad QAx = 0$$

which gives

$$A(x+y) = (PA+QA)(x+y) = PAx + QAy = Pb + Qb = b.$$

For the second clause, $PAy = 0$, it is enough that $y \in A^{-1}E$. For the fourth-clause, $QAx = 0$, is sufficies that $E^tAx = 0$.

Let $d = (E^tE)^{-1}E^tb$ and let $y = A^{-1}Ed$ (that's the second clause), then $QAy = QEd = Qb$: the second postulate.

If $y \in A^{-1}E$ then $Ay \in E \Rightarrow PAy = 0$, so

$$PAx = Pb \Rightarrow PA(x+y) = Pb.$$

The equation $PAx = Pb$ can be solved with a deflated method. I think.

Also, if $E^tAx = 0$ then $QAx = 0$, so

$$QAy = Qb \Rightarrow QA(x+y) = QB.$$

16.6.3 approach 3

Let N be a null-space, that is, $AN = 0$. Define $P = I - N(N^tN)^{-1}N^t$, then

$$\begin{cases} x = Ny \Rightarrow Px = 0 \\ N^tx = 0 \Rightarrow Px = x \end{cases}$$

Since $N^tP = 0$, $Px \perp N$. From $PN = 0 \Rightarrow N^tP^t = 0$ we get $P^tx \perp N$. Therefore, for all x, $P^tAPx \perp N$.

Same argument about Krylov space staying orthogonal to the null space.

Now solve $P^tAP(P^tx) = P^tb$ iteratively.

16.6.4 Null spaces and preconditioners

Let N be the right nullspace: $AN = 0$. Consider the fact that

$$x_{n+1} = x_0 + \sum_{i \le n} \alpha_i (M^{-1}A)^i r_0,$$

16.6.5 approach 4

Taken from [50].

Let N be the left nullspace of A: $N^tA = 0$. From $Ax = b$ this implies that $N^tb = 0$ needs to hold. Since the solution of singular a system is not unique, we impose a minimum length condition $N^tx = 0$.

We now consider a conjugate gradient method for solving $Ax = b$.

Lemma 76 *The residuals R and search directions P are N-orthogonal. Applying a CG process to $Ax = b$ is equivalent to iterating on the nonsingular system with coefficient matrix $\bar{Q}^tA\bar{Q}$.*

Proof: Augment N by $\bar{Q}$ ($N^t\bar{Q}=0$) such that $Q=[N,\bar{Q}]$ is unitary. The residual equation $R=AX-be^t$ gives $N^tR=0$ (note that no relations on X are used). Write

$$Q^tR=\begin{pmatrix}0\\ \bar{Q}^tR\end{pmatrix}$$

From $R=P(I-U)$ we get

$$Q^tP=\begin{pmatrix}0\\ \bar{Q}^tP\end{pmatrix}$$

Finally, $APD=R(I-J)$ becomes

$$\begin{array}{rcl}(Q^tAQ)(Q^tP)D & = & Q^tR(I-J)\\ \begin{pmatrix}0 & 0\\ 0 & \bar{Q}^tA\bar{Q}\end{pmatrix}\begin{pmatrix}0\\ \bar{Q}^tP\end{pmatrix}D & = & \begin{pmatrix}0\\ \bar{Q}^tR\end{pmatrix}(I-J)\end{array}$$

16.7 Shift invariance

For further reading: the material in this section is not referred to elsewhere.

Let A, R and H satisfy $AR=RH$, then also

$$(A-\alpha I)R=R(H-\alpha I).$$

To assuage some people's conscience we derive the finite form of this: let $AR_n=R_{n+1}H_{n+1n}$, then

$$\begin{array}{rcl}(A-\alpha I)R_n & = & R_{n+1n}H_{n+1n}-R_nI_n\alpha\\ & = & R_{n+1n}H_{n+1n}-R_{n+1}I_{n+1n}\alpha\\ & = & R_{n+1n}(H_{n+1n}-\alpha I)\end{array}$$

•

In principle one could use this fact to solve an indefinite system as follows:

- Compute $r_1=Ax_1-b$, and let $alpha$ be such that $A+\alpha I$ is definite.
- Build R and H that satisfy $(A+\alpha I)R=RH$.
- Construct iterates from $R=X(H-\alpha I)$.

This runs into the following objection: for an indefinite system the dominant components in the solution are likely to be those eigenvectors corresponding to eigenvalues close to zero. These become inner eigenvalues in the shifted system, and iterative methods typically find the outer eigenvalues better than the inner ones.

16.8 Equivalent formulas for the scalar quantities

For further reading: the material in this section is not referred to elsewhere.

In a parallel implementation, the conjugate gradient algorithm as formulated above has two synchronization points per iteration, located at the inner product calculations. These synchronization points cannot be coalesced immediately: the scalar d_{ii} needed to compute r_{i+1} needs an inner product with p_i, and the scalar u_{ii+1} necessary to compute p_{i+1} requires an inner product with r_{i+1}. (The basic structure of the algorithm is sketched, for reference purposes, in figure 9.) Thus the two inner products are inter-

---

- From p_i compute $p_i^t A p_i$; with $r_i^t M^{-1} r_i$, which was saved from the previous iteration, compute $\alpha_i = p_i^t A p_i / r_i^t M^{-1} r_i$.
- Update $r_{i+1} = r_i - \alpha_i A p_i$.
- Compute $r_{i+1} M^{-1} r_{i+1}$, and with it $\beta_i = r_{i+1} M^{-1} r_{i+1} / r_i M^{-1} r_i$.
- Update $p_{i+1} = M r_{i+1} - \beta_i p_i$.

Figure 9: Structure of inter-dependent inner products in the CG algorithm .

---

dependent. However, for the conjugate gradient method for spd systems the scalars can be computed in ways that eliminate one synchronization point, perhaps at the expense of some extra computation.

In this section we describe methods that reformulate one or the other of the inner products in such a way that the inner products can be computed simultaneously. At the moment, two complementary approaches are known. One can either eliminate the inner product $r_i^t r_i$ and express it in terms of inner products with the search directions (section 16.8.1), or one can try to eliminate the inner product $p_i^t A p_i$ and express it in terms of inner products involving gradients (section 16.8.2).

We will also describe other methods that do not follow this scheme, but that achieve the same aim of minimising the delay incurred by the inner products (section 16.8.3).

Combining inner products would seem to lead to a reduction of parallel runtime, as it halves the communication time of the global operations. However, Kaushik *et al* [46] note that the actual time in global operations is minimal, and that load balance is really to blame.

16.8.1 Elimination of the $r_i^t r_i$ inner product

The former approach was discovered by Saad [67]. From the orthogonality of R, the equation $APD = R(I-J)$ leads to

$$r_{i+1}^t M^{-1} r_{i+1} + r_i^t M^{-1} r_i = d_{ii}^2 (Ap_i)^t M^{-1} (Ap_i), \tag{158}$$

so the norms of r_i can be computed recursively without the need for an inner product. We give the resulting algorithm in figure 10. The balance of operations is:

- The preconditioner application $M^{-1} r_i$ is replaced by $M^{-1}(Ap_i)$; only in the first iteration do we compute $M^{-1} r_1$ explicitly, other $M^{-1} r_i$ vectors are obtained by vector updates.
- The $r_i^t M^{-1} r_i$ inner product is replaced by $(Ap_i)^t M^{-1} (Ap_i)$;

However, a simple analysis shows that this method is unstable, so extra measures are necessary. Hence, Meurant [54] proposed computing $r_i^t M^{-1} r_i$ explicitly, together with $(Ap_i)^t M^{-1} (Ap_i)$. Thus, the $r_{i+1}^t M^{-1} r_{i+1}$ value computed by equation (158) serves only as a predictor; it is later computed exactly. The resulting method (figure 11) takes three inner products per iteration, and is as stable as the classical formulation of the conjugate gradient method.

Question: *Would it make the method more stable if $M^{-1} r_{i+1}$ is computed explicitly, at the cost of an extra preconditioner application?*

- From p_i form Ap_i and $M^{-1}Ap_i$.
- Now compute simultaneously the inner products

$$p_i^t Ap_i \qquad \text{and} \qquad (Ap_i)^t M^{-1}(Ap_i);$$

with $r_i^t M^{-1} r_i$, which was saved from the previous iteration, compute $\alpha_i = p_i^t Ap_i / r_i^t M^{-1} r_i$, and the value of the inner product

$$r_{i+1}^t M^{-1} r_{i+1} = -r_i M^{-1} r_i + \alpha_i^2 (Ap_i)^t M^{-1}(Ap_i).$$

With this, also compute $\beta_i = r_{i+1} M^{-1} r_{i+1} / r_i M^{-1} r_i$. Save the vector $M^{-1}Ap_i$.
- Update $M^{-1}r_{i+1} = M^{-1}r_i - \alpha_i M^{-1}Ap_i$.
- Update $p_{i+1} = Mr_{i+1} - \beta_i p_i$.

Figure 10: Saad's CG method with one synchronisation point.

16.8.2 Elimination of the $p_i^t Ap_i$ inner product

Recently, several methods based on elimination of computing $p_i^t Ap_i$ (which is needed for $d_{ii} = r_i^t M^{-1} r_i / p_i^t Ap_i$) have been discovered. These methods combine $R^t M^{-1} AP = (I-U)^t P^t AP$ and $R^t M^{-1} AR = R^t AP(I-U)$ to

$$R^t M^{-1} AM^{-1} R = (I-U)^t P^t AP(I-U)$$
$$\Rightarrow P^t AP = R^t M^{-1} AM^{-1} R + P^t APU + U^t P^t AP - U^t P^t APU.$$

Using the A-orthogonality of P and the fact that the second and third term in the rhs are strictly upper and lower triangular respectively, we find that

$$p_i^t Ap_i = r_i^t M^{-1} AM^{-1} r_i - u_{i-1i}^2 p_{i-1}^t Ap_{i-1}. \tag{159}$$

For the resulting variant of the conjugate gradient method Ap_i is again computed recursively from $AM^{-1}r_i$. The inner products $r_i^t M^{-1} r_i$ and

- From p_i form Ap_i and $M^{-1}Ap_i$.
- Now compute simultaneously the inner products

$$p_i^t Ap_i, \qquad r_i^t M^{-1} r_i, \qquad \text{and} \qquad (Ap_i)^t M^{-1}(Ap_i);$$

- with $r_i^t M^{-1} r_i$ compute $\alpha_i = p_i^t Ap_i / r_i^t M^{-1} r_i$, and the value of the inner product $r_{i+1}^t M^{-1} r_{i+1} = -r_i M^{-1} r_i + \alpha_i^2 (Ap_i)^t M^{-1}(Ap_i)$. With this, also compute $\beta_i = r_{i+1} M^{-1} r_{i+1} / r_i M^{-1} r_i$. Save the vector $M^{-1}Ap_i$.
- Update $r_{i+1} = r_i - \alpha_i Ap_i$ and $M^{-1}r_{i+1} = M^{-1}r_i - \alpha_i M^{-1}Ap_i$.
- Update $p_{i+1} = M^{-1}r_{i+1} - \beta_i p_i$.

Figure 11: Meurant's modification of Saad's CG method.

$r_i^t M^{-1} AM^{-1} r_i$ are computed simultaneously, and the scalar $p_i^t Ap_i$, needed for $d_{ii} = r_i^t M^{-1} r_i / p_i^t Ap_i$, is computed from the above recurrence. This method was proposed by Chronopoulos and Gear [14], and later reinvented by the present author and [20], with a full analysis in [19]. The algorithm is given in figure 12.

There is a fairly strong heuristic argument for the stability of this last rearrangement: using equations (80), (81) and (64) the recurrence for $p_n^t Ap_n$ can be derived from the recurrence

$$d_{nn}^{-1} = h_{n+1n+1} - h_{n+1n} d_{nn} h_{nn+1}$$

for pivot generation in the factorization of the Hessenberg matrix. For symmetric positive definite systems this is a stable recurrence, see the analysis in [19].

On the other hand, we note that the original CG algorithm had a fairly good cache behaviour, in that vectors were used pretty much immediately after they were generated. This rearranged algorithm has a far less

- From r_i, $M^{-1}r_i$, and $AM^{-1}r_i$ compute simultaneously

$$r_i^t M^{-1} r_i \qquad \text{and} \qquad (M^{-1}r_i)^t A M^{-1} r_i;$$

- with this, compute $\beta_i = r_{i+1}M^{-1}r_{i+1}/r_i M^{-1} r_i$.
- Recursively compute $p_i^t A p_i$ from $r_i^t M^{-t} A M^{-1} r_i$ and $p_{i-1}^t A p_{i-1}$ using the formula

$$p_i^t A p_i = r_i^t M^{-t} A M^{-1} r_i - \beta_i^2 p_{i-1}^t A p_{i-1},$$

- With $p_i^t A p_i$ compute $\alpha_i = p_i^t A p_i / r_i^t M^{-1} r_i$, and use this to update x_i and r_i to x_{i+1} and r_{i+1}.
- Apply the preconditioner to form $M^{-1}r_{i+1}$, then the matrix to form $AM^{-1}r_{i+1}$.
- Now construct Ap_{i+1} recursively by

$$Ap_{i+1} = AM^{-1}r_{i+1} + Ap_i\beta_i$$

Figure 12: Chronopoulos and Gear's method

localised behaviour.

Another variant of this scheme, also proposed by Eijkhout [24], results from considering the diagonal of

$$(I-U)^t PAP = R^t AP = R^t AR(I-U)^{-1}.$$

Since R^tAR is tridiagonal the infinite expansion of $(I-U)^{-1}$ terminates quickly, and we find that

$$p_i^t A p_i = r_i^t A r_i + r_i^t A r_{i-1} u_{i-1i}.$$

However, in the presence of rounding errors, this method, being based on an infinite number of orthogonalities, becomes unstable in contrast to the two previous methods which only use a single orthogonality relation.

Methods for removing a synchronization point generalize to the Lanczos method. We find that

$$q_{i+1}^t A p_{i+1} = s_{i+1}^t A r_{i+1} - s_i^t A p_i u_{ii+1}$$

from $(I-U)^t Q^t AP = S^t AR + S^t APU$;

$$q_{i+1}^t A p_{i+1} = s_{i+1}^t A r_{i+1} - u_{ii+1}^2 q_i^t A p_i$$

using A-orthogonality of Q and P in

$$Q^t AP = S^t AR + Q^t APU + U^t Q^t AP - U^t Q^t APU,$$

and we get

$$s_{i+1}^t r_{i+1} - s_i^t r_i = d_{ii}^2 (A^t q_i)^t (Ap_i)$$

from the equations $APD = R(I-J)$ and $A^tQD = S(I-J)$ using orthogonality of S and R.

16.8.3 Other approaches for minimizing synchronisation overhead

An approach for a reduced synchronization overhead was proposed by [78] and [14], based on computing a number of Krylov vectors and orthogonalizing these as a block. This approach requires some caution since it is less stable. See also section 11.3.2.

Van der Vorst, in Demmel *et al.* [22], proposed another reformulation CG method that eliminates synchronisation overhead. This method, which unlike the above ones can not be generalized to the nonsymmetric case, is based on the following observation. If M is symmetric and factored as LL^t, the inner product $r^tM^{-1}r$ can be computed as $(L^{-1}r)^t(L^{-1}r)$, that is, the value of the inner product can be computed after only the forward solve

has been performed. Therefore, the computation of the backward solve, taking L^{-t} times the intermediate result $L^{-1}r$, can overlap the communication involved in the inner product of $(L^{-1}r)^t(L^{-1}r)$.

Note that this idea presupposes that M is both a factorisation, and does not involve communication at all or at least less than for an inner product. Examples would be the Block Jacobi method or an additive Schwarz method, where in both cases the local solve is realised through a Cholesky factorisation.

In order to accomodate as much communication delay as possible, we also delay the solution update slightly, to let it also happen during the inner product communication stage. As a result, the normal sequence

update $r_{i+1} = r_i - \alpha_i A p_i$
$x_{i+1} = x_i - \alpha_i p_i$
solve $z_{i+1} = M^{-1} r_{i+1}$
compute $\beta_{i+1} = z_{i+1}^t r_{i+1}$

now becomes

update $r_{i+1} = r_i - \alpha_i A p_i$
solve $s_{i+1} = L^{-1} r_{i+1}$
compute the local parts of $s_{i+1}^t s_{i+1}$ and
start an asynchronous reduction to sum them
meanwhile:
update $x_{i+1} = x_i - \alpha_i p_i$
solve $z_{i+1} = L^{-t} s_{i+1}$
wait for the reduction to finish,
compute $\beta_{i+1} = z_{i+1}^t r_{i+1}$

This method is fully equivalent to the original CG algorithm, except for the slightly different calculation of the $z^t r$ inner product. In fact, it is slightly more stable because of the symmetric nature of the inner product calculation.

This method still has one inner product that forms a bottleneck. We can eliminate this by combining van der Vorst's above approach with Saad's method (or rather, Meurant's modification of it) from section 16.8.1. We proceed as follows:

generate r_{i+1} and $z_{i+1} = M^{-1} r_{i+1}$ by update relations
solve $q_{i+1} = L^{-1} A p_{i+1}$
compute the local parts of $z_i^t r_i$, $p_i^t A p_i$, $q_{i+1}^t q_{i+1}$, and
start an asynchronous reduction to sum them
meanwhile:
update $x_{i+1} = x_i - \alpha_i p_i$
solve $L^{-t}(L^{-1} A p_i)$
wait for the reduction to finish.

Another way of hiding communication is to delay the update of the solution vector and let this overlap one of the inner products.

16.8.4 Pipelined CG

The overlapping approach of van der Vorst also underlies the 'pipelined' iterative methods. For instance, pipelined CG is based on introducing explicit recurrences for $M^{-1}r_i$ and Ap_i:

$$\text{let } q_1 = Ap_1, z_1 = M^{-1} r_1 \text{ be given}$$

$$\begin{cases} \pi = p_1^t A p_1 \\ \alpha = \rho_1/\pi \\ r_2 = r_1 + A p_1 \alpha \\ z_2 = M^{-1} r_2 \\ \rho_2 = r_2^t z_2 \\ \beta = \rho_2/\rho_1 \\ p_2 = z_2 + p_1 \beta \end{cases} \Rightarrow \begin{cases} \pi = p_1^t q_1 \\ \alpha = \rho_1/\pi \\ \begin{cases} r_2 = r_1 + q_1 \alpha \\ z_2 = z_1 + M^{-1} q_1 \end{cases} \\ \rho_2 = r_2^t z_2 \\ \beta = \rho_2/\rho_1 \\ \begin{cases} p_2 = z_2 + p_1 \beta \\ q_2 = A z_2 + q_1 \beta \end{cases} \end{cases}$$

We now observe that the added relations contain terms $M^{-1}q_1$ and Az_2 that can be overlapped with inner product calculations:

$$
\begin{cases}
\text{overlapped:} \\
\text{compute } M^{-1}q_1 \\
\pi = p_1^t q_1
\end{cases}
$$
$$\alpha = \rho_1/\pi$$
$$
\begin{cases}
\text{in arbitrary order:} \\
r_2 = r_1 + Ap_1\alpha \\
z_2 = z_1 + M^{-1}q_1\alpha
\end{cases}
$$
$$
\begin{cases}
\text{overlapped:} \\
\text{compute } Az_2 \\
\rho_2 = r_2^t z_2
\end{cases}
$$
$$\beta = \rho_2/\rho_1$$
$$
\begin{cases}
\text{in arbitrary order:} \\
p_2 = z_2 + p_1\beta \\
q_2 = Az_2 + q_1\beta
\end{cases}
$$

16.9 Complex symmetric CG

We recall lemma 47 which states that $AR = RH$ is a tri-diagonalisation of A if A is symmetric. It is easy to see that this statement also holds for complex symmetric systems. Thus, if we apply a CG algorithm to complex a symmetric system, but we use the $x^t y$ inner product[23] instead of the $x^* y$ one, we arrive at a three-term recurrence for the residual vectors.

The straightforward generalisation of the CG algorithm to complex spaces would use the $x^* y$ inner product, and would only yield three-term recurrences for Hermitian systems. General complex non-Hermitian systems would need long recurrences, as in the case of Arnoldi orthogonalisation methods for nonsymmetric real systems.

23. Which is not strictly speaking an inner product in complex spaces.

16.10 Double size systems

There are several observations to be made about the augmented matrix

$$\begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix}.$$

16.10.1 Singular value computations

Let P be normalized orthogonal combinations of a Krylov sequence $K\langle AA^t, p_1\rangle$, so that the Hessenberg matrix in $AA^tP = PH$, which is tridiagonal because of the symmetry of AA^t, can be written as $H = BB^t$ with B upper bidiagonal (see lemma 48). Introducing a sequence Q that satisfies $AQ = PB$ we get a coupled system

$$AQ = PB, \quad A^tP = QB^t.$$

Lanczos in [49] observed that this coupled system could be generated by iterating on a double size matrix: the resulting system is

$$\begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix} \begin{pmatrix} P & 0 \\ 0 & Q \end{pmatrix} = \begin{pmatrix} P & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} 0 & B \\ B^t & 0 \end{pmatrix}.$$

The equation $AA^tR = RBB^t$ can be used to compute singular values: if R is unitary ($R^t = R^{-1}$; Conjugate Gradient iteration yields an orthogonal R, which can be made unitary by scaling), then AA^t and BB^t have the same eigenvalues, hence A and B have the same singular values. Consequently, the double size iteration, or equivalently the coupled relations for P and Q, can be used as a preliminary step of bi-diagonalisation in computing singular values. This method is used by Golub, Luk, and Overton [36]; they use a block method which turns B into a block bidiagonal matrix where the diagonal blocks are upper triangular, and the upper diagonal blocks are lower triangular.

See also section 16.10.3.

16.10.2 Iterating on the normal equations; LSQR

The Conjugate Gradient method only works for symmetric (and positive definite) systems. Hence, people have advocated solving nonsymmetric systems by iterating on the normal equations, that is, replacing $Ax = b$ by

$$A^tAx = A^b.$$

We can split this equation as

$$\left.\begin{matrix} y = Ax \\ A^ty = A^tb \end{matrix}\right\} \Rightarrow \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix} \begin{pmatrix} y & x \end{pmatrix} = \begin{pmatrix} 0 & A^tb \end{pmatrix}.$$

This seems to be the basis of the LSQR method of Paige and Saunders [64].

16.10.3 Lanczos iteration as augmented Conjugate Gradient iteration

Lanczos [49] observed that the left and right sequences of the Lanczos iteration can be generated from a single Conjugate Gradient iteration process with double size vectors. Specifically, it uses as iteration matrix

$$\mathbf{A} = \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix}.$$

Let R and S be the right and left sequences, that is, based on the Krylov sequences for A and A^t, and P and Q the corresponding search directions. We express the augmented iteration in terms of the sequences

$$\mathbf{p}_i = \begin{pmatrix} q_i \\ p_i \end{pmatrix}, \quad \mathbf{r}_i = \begin{pmatrix} r_i \\ s_i \end{pmatrix},$$

and we use a preconditiong matrix

$$\mathbf{M}^{-1} = \begin{pmatrix} 0 & M^{-t} \\ M^{-1} & 0 \end{pmatrix};$$

we note that the matrix and preconditioner are symmetric.

The matrix vector product is

$$\mathbf{A}\mathbf{p}_i = \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix} \begin{pmatrix} q_i \\ p_i \end{pmatrix} = \begin{pmatrix} Ap_i \\ A^t q_i \end{pmatrix},$$

and the preconditioner application is

$$\mathbf{M}^{-1}\mathbf{r}_i = \begin{pmatrix} 0 & M^{-t} \\ M^{-1} & 0 \end{pmatrix} \begin{pmatrix} r_i \\ s_i \end{pmatrix} = \begin{pmatrix} M^{-t}s_i \\ M^{-1}r_i \end{pmatrix}.$$

The $q_i^t A p_i$ inner product becomes

$$\mathbf{p}_i^t \mathbf{A}\mathbf{p}_i = \begin{pmatrix} q_i^t & p_i^t \end{pmatrix} \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix} \begin{pmatrix} q_i \\ p_i \end{pmatrix} = q_i^t A p_i + p_i^t A^t q_i = 2 q_i^t A p_i,$$

and the $s_i M^{-1} r_i$ inner product becomes

$$\mathbf{r}_i^t \mathbf{M}^{-1}\mathbf{r}_i = \begin{pmatrix} r_i^t & s_i^t \end{pmatrix} \begin{pmatrix} 0 & M^{-t} \\ M^{-1} & 0 \end{pmatrix} \begin{pmatrix} r_i \\ s_i \end{pmatrix} = r_i^t M^{-t} s_i + s_i^t M^{-1} r_i = 2 s_i^t M^{-1} r_i.$$

However, in both inner products we use the fact that scalars can be transposed, that is, this method will not work as a block method.

Question: *Reconcile this with the fact that Golub et al use this trick for a block method.*

The residual update

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{A}\mathbf{p} \frac{\mathbf{r}_i^t \mathbf{M}^{-1} \mathbf{r}_i}{\mathbf{p}_i^t \mathbf{A} \mathbf{p}_i}$$

comprises both the left and right residual update, and likewise does the search direction update

$$\mathbf{p}_{i+1} = \mathbf{M}^{-1}\mathbf{r}_{i+1} - \mathbf{p}_i \frac{\mathbf{r}_{i+1}^t \mathbf{M}^{-1} \mathbf{r}_{i+1}}{\mathbf{r}_i \mathbf{M}^{-1} \mathbf{r}_i}.$$

If we compute the Lanczos sequences by three-term recurrences (section 8.3) we need to compute inner products such as $s_i^t M^{-1} A M^{-1} r_i$. For the augmented vectors:

$$\mathbf{r}_i^t \mathbf{M}^{-1} \mathbf{A} \mathbf{M}^{-1} \mathbf{r}_i = \begin{pmatrix} s_i^t M^{-1} & r_i^t M^{-t} \end{pmatrix} \begin{pmatrix} A M^{-1} r_i \\ A^t M^{-t} s_i \end{pmatrix}$$

$$= s_i^t M^{-1} A M^{-1} r_i + r_i^t M^{-t} A^t M^{-t} s_i = 2 s_i^t M^{-1} A M^{-1} r_i.$$

16.11 Scaled sequences

For further reading: the material in this section is not referred to elsewhere.

If V is an arbitrary sequence satisfying $AV = V\tilde{H}$ with $\tilde{H}$ an upper Hessenberg matrix, then V can sometimes be scaled to a residual sequence R. We first give the algorithm, then discuss the conditions under which it is well-defined.

Algorithm: Scale combination of Krylov sequence to residual sequence

Algorithm 10 *Given $AV = V\tilde{H}$, construct a diagonal matrix Ω such that $R = V\Omega$ is a residual sequence.*

Define $H = \Omega^{-1}\tilde{H}\Omega$, then $AR = RH$, and the algorithm is equivalent to constructing Ω such that H, by theorem 30, has zero column sums.

The choice for ω_1 is arbitrary, or determined by the context. For $n = 1, 2, \ldots$:

1. *Let ω_k be given for $k \leq n$. This is true for $n = 1$.*
2. *For $k < n$, $h_{kn} = \omega_k^{-1}\tilde{h}_{kn}\omega_n$ can be computed.*
3. *Define $h_{nn} = \tilde{h}_{nn}$.*
4. *The zero column sum requirement lets us compute $h_{n+1n} = -\sum_{k\leq n} h_{kn}$; from $h_{n+1n} = \omega_{n+1}^{-1}\tilde{h}_{n+1n}\omega_n$ we can compute*

$$\omega_{n+1} = \tilde{h}_{n+1n}\omega_n / h_{n+1n}. \tag{160}$$

16.11.1 Theory

In this section we will show that orthogonal sequences satisfying a Hessenberg relation can be scaled to a residual sequence. We start by giving a lemma that translates equation (160) in terms of the sequence to be scaled.

Condition for scaling a sequence to a residual sequence

Lemma 77 *Let a matrix A, an irreducible upper Hessenberg matrix $\tilde{H}$, and a sequence V satisfying $AV = V\tilde{H}$ be given, then V can be scaled as $R = V\Omega$ to a residual sequence if and only if every v_i has a nonzero component of v_1.*

Proof: Let $K = K\langle A, v_1\rangle$, and suppose that V can be scaled to a residual sequence $R = V\Omega$. In particular, all $\omega_i \neq 0$. We then have

$$AV = V\tilde{H} \qquad \text{and} \qquad AR = R\Omega^{-1}\tilde{H}\Omega \equiv RH,$$

so there are nonsingular triangular matrices $U, \tilde{U}$ such that

$$V = K\tilde{U}, \qquad R = KU,$$

and they are related by $U = \tilde{U}\Omega$. The 'only if' half of the statement now follows simply: since $u_{1j} \equiv 1$ (see lemma 9) it follows that all $\tilde{u}_{1j} \neq 0$.

Conversely, suppose that, with $\tilde{U}$ as defined above, $\tilde{u}_{1j} \neq 0$. From the above algorithm we find that

$$\omega_{n+1} = \tilde{h}_{n+1n}\omega_n / h_{n+1n} = -\frac{\tilde{h}_{n+1n}\omega_n}{\sum_{k\leq n} h_{kn}} = -\frac{\tilde{h}_{n+1n}}{\sum_{k\leq n} \omega_k^{-1}\tilde{h}_{kn}},$$

Rewriting the sum in the numerator in terms of $\tilde{U}$, where we recall from equation (31) that $\tilde{H} = \tilde{U}^{-1}J\tilde{U}$, gives

$$\sum_{k\leq n} \omega_k^{-1}\tilde{h}_{kn} = -\tilde{u}_{1n+1}\tilde{u}_{nn}/\tilde{u}_{n+1n+1}.$$

Having $\tilde{u}_{1j} \neq 0$ (and note that $u_{nn} \neq 0$ from the irreducibility of H) is thus seen to be a sufficient condition for the above algorithm to be well-defined. •

As an example of a sequence that can not be scaled to a residual sequence, consider a Krylov sequence, that is, $H = J$. In this case the Hessenberg

matrix is not irreducible, so the condition of lemma 77 are not satisfied. As a more intuitive argument, note that the scaling would have to be such that H winds up with zero column sums, which is clearly impossible.

In the following theorem we prove that the $\tilde{u}_{1j} \neq 0$ condition is always satisfied if V is an orthogonal sequence.

An orthogonal sequence can be scaled to a residual sequence

Theorem 78 *Let a non-singular matrix matrix A, an irreducible upper Hessenberg matrix H, and a sequence V satisfying $AV = VH$ be given. If V is orthogonal, it can be scaled to a residual sequence.*

Proof: Let V be an orthogonal sequence, let K be the Krylov sequence

$$K = K\langle A, \alpha v_1 \rangle, \qquad \text{some } \alpha \neq 0,$$

and let U be a non-singular upper triangular matrix such that $V = KU$. Suppose by contradiction that there is a smallest index n for which $u_{1n} = 0$. From the above we recall that the existence of such an index precludes the sequence from being scaled to a residual sequence.

We will now investigate the consequences of the identity

$$v_n = \sum_{i=1}^{n} k_i u_{in}$$

which is the n-th column of $V = KU$. Using $u_{1n} = 0$, elementary rewriting gives

$$v_n = \sum_{i=1}^{n} k_i u_{in} = \sum_{i=2}^{n} k_i u_{in} = \sum_{i=2}^{n} A k_{i-1} u_{in} = A \sum_{i=1}^{n-1} k_i u_{i+1n}. \tag{161}$$

From the non-singularity of U we can express each k_i as a linear combination of v_j vectors ($j \leq i$), so equation (161) translates to the statement that there are coefficients α_i such that

$$v_n = A \sum_{i=1}^{n-1} v_i \alpha_i$$

From the orthogonality of V we then find that

$$v_n^t A^{-1} v_n = v_n^t \sum_{i=1}^{n-1} v_i \alpha_i = \sum_{i=1}^{n-1} 0 \cdot \alpha_i = 0,$$

and from the non-singularity of A this implies that $v_n = 0$. Since this contradicts the orthogonality of V, we find that $u_{1n} \neq 0$. Hence, by lemma 77, we find that the sequence V can be scaled to a residual sequence. •

Question: *How do you reconstruct the α, β coefficients of the original algorithm?*

The condition of the irreducibility of H in the above results is essential for iterative methods.

Remark 79 *Suppose that $h_{n+1n} = 0$, then $AV_n = V_n H_n$, so V_n, and consequently the Krylov sequence K_n from which it was built, form an invariant subspace.*

Thus, a reducible Hessenberg matrix precludes forming more accurate approximations to the solution of the linear system under consideration. See section 14.4 for more details.

16.11.2 A normalised CG algorithm

We will now derive a CG algorithm that uses normalised residuals. There is a point to such methods. For instance, suppose that the matrix-vector product is not an explicitly given operation, but (in the context of a non-linear solver) evaluated as

$$F(u)v \approx (f(u+hv) - f(u))/h \qquad \text{some fixed } h,$$

then the error is proportional to $||v||$, so it pays to normalise, or at least limit the size of, the vectors that are multiplied.

In particular, we will derive here a variant of the Conjugate Gradient algorithm with coupled two-term recurrences, that generates an orthonormal sequence which later will be scaled to a residual sequence. This is of course reminiscent of the GMRES algorithm which also generates an orthonormal sequence. However, that sequence is never explicitly scaled to a residual sequence.

Consider the usual equations for a conjugate gradient method

$$APD = R(I-J), \qquad M^{-1}R = P(I-U), \qquad R^tM^{-1}R = \Omega^2.$$

We will now derive an algorithm that directly generates the normalised vectors

$$V = R\Omega^{-1}.$$

With

$$Q = P\Omega^{-1}, \qquad \tilde{U} = \Omega U\Omega^{-1}, \qquad L = \Omega J\Omega^{-1},$$

it is easy to check that we get transformed relations (and note that D is not transformed)

$$M^{-1}V = Q(I-\tilde{U}), \qquad AQD = V(I-L).$$

Replacing $V \rightarrow R$, et cetera, we now get the defining relations

$$APD = R(I-L), \qquad MR = P(I-U), \qquad R^tM^{-1}R = I. \tag{162}$$

First of all, the coefficients of L follow from the normalisation, since

$$r_{i+1}\ell_{i+1i} = r_i - Ap_id_i.$$

From the second relation of equation (162) we get

$$R^tM^{-1}R = R^tP(I-U)$$

so P^tR is lower triangular with identity diagonal. From the first relation we get

$$P^tAPD = P^tR(I-L)$$

so P^tAP is lower triangular and

$$d_i = 1/p_i^tAp_i.$$

Combining the first two relations we get

$$\begin{aligned} R^tM^{-t}AP &= R^tM^{-t}R(I-L)D^{-1} \\ &= (I-U)^tP^tAP \end{aligned}$$

so

$$P^tA^tP(I-U) = D^{-1}(I-L^t).$$

In this last equation we consider column $n+1$:

$$\begin{pmatrix} p_1^tA^tp_1 & \cdots & p_1^tA^tp_{n+1} \\ & \ddots & \vdots \\ & & p_{n+1}^tA^tp_{n+1} \end{pmatrix} \begin{pmatrix} -u_{1n+1} \\ \vdots \\ -u_{nn+1} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -d_n^{-1}\ell_{n+1n} \\ d_{n+1}^{-1} \end{pmatrix} \tag{163}$$

This first of all confirms the value for d_{ii} derived above. In general, we can now solve the $n+1$st column of U; in the particular case of symmetry we get $u_{nn+1} = \ell_{n+1n}$.

The one issue we have not tackled in this normalised algorithm is the solution update. Remembering that the true search direction is ω_i times the normalised one, we find the solution update to be

$$x_{i+1} = x_i + p_id_i\omega_i.$$

We now give the full preconditioned algorithm for the symmetric case.

Algorithm 11 *Let x_1, f, A be given, and compute*

$$r_1 = Ax_1 - f, z_1 = M^{-1}r_1, \qquad \omega_1 = \|r_1\|, \qquad r_1 \leftarrow r_1/\omega_1, z_1 \leftarrow z_1/\omega_1.$$

Now iterate for $i = 1, \ldots$:

- *Perform a stopping test on $r_i\omega_i$.*
- *For $i = 1$, $p_1 = z_1$; in general $u_{i-1i} = \omega_{i-1}$, and*

$$p_i = z_i + p_{i-1}\omega_{i-1}.$$

- *Let $d_i = 1/p_i^t A p_i$; use this to update:*

$$x_{i+1} = x_i - p_i d_i \omega_i, \qquad r_{i+1} = r_i - A p_i d_i, z_i = M^{-1} r_i.$$

- *Subsequently we normalise the residual:*

$$\ell_{i+1i} = \sqrt{z_{i+1}^t r_{i+1}}; \qquad r_{i+1} \leftarrow r_{i+1}/\ell_{i+1i}, z_{i+1} \leftarrow z_{i+1}/\ell_{i+1i}.$$

- *Construct column i of the Hessenberg matrix $\tilde{H} = (I-L)D(I-U)$ of the normalised sequence, and derive H from it as described in algorithm 10; construct ω_{i+1}.*

In this algorithm we normalised the r_i vectors, while the matrix-vector product is performed on the p_i vectors. While these are not themselves normalised, we expect them to be very much limited in norm, given their close relation to the r_i vectors. It would be possible to formulate the algorithm such that not $\|r_i\| = 1$, but $\|p_i\|_A = 1$ is satisfied.

This attempt is in the right direction:

```
function [x,its] = cgs(A,M,Minit,Mapply,b,red,maxit);
%  function [x,its] = cgs(A,M,Minit,Mapply,b,red,maxit);
%
% Conjugate Gradient Squared method using normalised
% residual vectors.
% The preconditioner M is solved by "eval(Mapply);"
% where Mapply is a string computing 'z' from 'r';
% empty string allowed. Example: Mapply == "z = M.*r".
% Minit is also a string, again empty is allowed.

% setup preconditioner
if ~isempty(Minit), eval(Minit); end;

its = [];
x = 0*b;
r1 = A*x-b;
if ~isempty(Mapply); r = r1; eval(Mapply); rr = z;
  else, rr = r1; end;
Dinv = [];
om = sqrt(r1'*rr); rr = rr/om^2; Om = [om];

for it=1:maxit,
om
  Un(it,it) = 1;
  if it>1,
    beta = el; Un(it-1,it) = -el; end;
% 3 = 2 + 1 beta => pr = rr + pr1 beta
  if it>1,
    pr = rr + pr1*beta; else, pr = rr; end;
```

```
% 4 = 3 + (1+ 4 beta) beta => pp = pr + (pr1+pp beta)beta
  if it>1,
    pp = pr + (pr1+pp*beta)*beta; else pp = pr; end;
% MinvA 4 => mapp = M \ (A*pp)
  if ~isempty(Mapply), r = A*pp; eval(Mapply); mapp = z;
    else; mapp=A*pp; end;
% p'*A*p dot product
  pap_dot = r1'*mapp; alpha = 1/pap_dot;
  Dinv = [Dinv,pap_dot];
% 1 = 3 - MinvA 4 alpha => pr1 = pr - M \ A pp alpha
  pr1 = pr - mapp*alpha;
% 2 = 2 - Minv A ( 3 alpha_r + 1 alpha_l )
% => rr = rr - M \ A ( pr + pr1 ) alpha
  search = (pr+pr1)*alpha;
  x = x-search;
  if ~isempty(Mapply), r = A*search; eval(Mapply); rr = rr - z;
    else, rr = rr- A*search; end;
% normalisation
  el = sqrt(r1'*rr); rr = rr/el^2;
  Ln(it,it) = 1; Ln(it+1,it) = -el;
% Hessenberg matrices
  Hn = Ln*diag(Dinv)*Un;
  H(1:it-1,it) = Om(1:it-1)'.\Hn(1:it-1,it)*Om(it);
  H(it,it) = Hn(it,it); H(it+1,it) = -sum(H(1:it,it));
  om = Hn(it+1,it)*om/H(it+1,it); Om = [Om,om];
end;

fprintf('Final norm: %e\n',norm(A*x-b));
```

Note:

- The `rr` vector is divided by ℓ^2, so that the r_1^t`rr` product comes out as one.
- The $\boxed{2}_i$ update is probably wrong because we don't normalise the left r_i sequence. Maybe reconstruct the α and *beta* coefficients and do the left sequence exact?

16.12 Seed systems

For further reading: the material in this section is not referred to elsewhere.

In several sections of this monograph we have seen block methods, where multiple systems with identical coefficient matrix but different righthand sides are solved. It is also possible to reuse information from one system if the next system has the same coefficient matrix, or a sufficiently similar one.

Chan and Ng [13] consider the case of not identical coefficient matrices. Their convergence theorem relates the true solution x of a second system to the approximate one $\bar{x}$ obtained by projecting onto the subspace K_m from the first system as

$$x^{(2)} - \bar{x}^{(2)} = \sum_k c_k q_k, \qquad |c_k| \leq E_k + F$$

where q_k are eigenvectors of the second system and

$$E_k = \|P_m^{\perp} x^{(2)}\|_2 \, |\sin \angle(q_k, K_m)|$$

and

$$F = \|{A^{(1)}}^{-1}\|_2 \, \|(A^{(2)} - A^{(1)})x^{(2)}\|_2$$

where V_m is the orthonormal scaling of K_m, $P_m^{\perp} = I - V_m T_m^{-1} V_m^t A^{(1)}$ is the $A^{(1)}$-orthogonal projection onto K_m (see section 7.7 for the theory of projection in iterative methods), and $T_m = V_m^t A^{(1)} V_m$.

16.13 Polynomial acceleration

For further reading: the material in this section is not referred to elsewhere.

Methods such as conjugate gradients are sometimes considered to be an acceleration of a basic iterative method. Suppose iterates x_n have been generated by (**??**), then the idea behind acceleration is to take combinations

$$\tilde{x}_n = \sum_{i=1}^{n} x_i u_{in} \tag{164}$$

with the consistency condition

$$\sum_{i=1}^{n} u_{in} = 1. \tag{165}$$

Varga [**?**] calls (164) the semi-iterative method corresponding to the basic iterative method. The residuals $\tilde{r}_n = A\tilde{x}_n - f$ are seen to satisfy $\tilde{R} = RU$ (where U is an upper triangular matrix containing the u_{in} coefficients) because of the above consistency condition.

Writing the basic iterative method as $AR = R(I - J)$ we find for the accelerated residuals $A\tilde{R} = \tilde{R}H$ where H is the upper Hessenberg matrix $I - U^{-1}JU$.

Lemma 80 *The consistency condition (165) implies the consistency condition of lemma 23 for $H = I - U^{-1}JU$.*

Proof: We can express equation (165) as $e^t U = e^t$. Since this implies $e^t U^{-1} = e^t$, and since $e^t J = e^t$, we find that $e^t H = 0$. •

These acceleration methods are also called 'polynomial acceleration' methods because of the following fact.

Lemma 81 *The accelerated residuals are related to the original residuals as*

$$\tilde{r}_n = P_n(I-A)r_1$$

where P_n is an $n-1$-st degree polynomial with its coefficients in the n-th column of U, where U contains the u_{in} coefficients of equation (164).

Proof: Lemma **??** established that the residuals of the basic method are Krylov vectors in a sequence with matrix $I-A$. It then follows from lemma 7 that the combinations $\tilde{R}$ are obtained by multiplying by a matrix polynomial as stated above. •

16.14 Left and right preconditioning

For further reading: the material in this section is not referred to elsewhere.

In this monograph we have derived all methods in their 'preconditioned' form, that is, involving a matrix M. The substitution $M = I$ in the methods derived above gives the 'unpreconditioned' version, which is traditionally the one derived first. Deriving a preconditioned version could then be done by substituting either AM^{-1} or $M^{-1}A$ for A in the unpreconditioned method. This gives rise to two methods:

$$\text{Left preconditioned method: solve } M^{-1}Ax = M^{-1}b. \tag{166}$$

and

$$\text{Right preconditioned method: solve } AM^{-1}x' = b;\ x = M^{-1}x'. \tag{167}$$

and a combination two-sided preconditioning method: $M_1^{-1}AM_2^{-1}x' = M_1^{-1}b$, $x = M_2^{-1}x'$.

These methods are equivalent to the normal form derived above, as we will show now. In particular, we address these two points:

- Is a one-sided preconditioner equivalent to (symmetric) Conjugate Gradients for a combination of symmetric matrix and symmetric preconditioner? and,
- Are the solution and residual corresponding to the original problem easily available in each iteration, without spending too much work on back transformations?

16.14.1 Left preconditioning

The left-preconditioned BiCG method (which we take as prototypical of Krylov space methods) is derived by applying an unpreconditioned algorithm to the coefficient matrix $M^{-1}A$, the initial guess x_1, and the right hand side $M^{-1}b$.

Algorithm: Left preconditioned BiConjugate Gradient Method

Algorithm 12 *Let x_1, A, b, and M be given. Set initially $r_1 = M^{-1}Ax_1 - M^{-1}b$. Then, in each iteration:*

$$\begin{aligned} x_{n+1} &= x_n + p_n\alpha_n \\ r_{n+1} &= r_n + M^{-1}Ap_n\alpha_n \\ s_{n+1} &= s_n + A^tM^{-t}q_n\alpha_n \\ & \quad \text{where } \alpha_n = (s_n^t r_n)/(q_n^t M^{-1}Ap_n) \end{aligned}$$

and

$$\begin{aligned} p_{n+1} &= r_{n+1} - p_n\beta_n \\ q_{n+1} &= s_{n+1} - q_n\beta_n \\ & \quad \text{where } \beta_n = (s_{n+1}^t r_{n+1})/(s_n^t r_n) \end{aligned}$$

We rewrite, expressing the algorithm in terms of transformed quantities Mr_n and $M^{-t}q_n$:

$$\begin{aligned} Mr_1 &= Ax_1 - b \\ x_{n+1} &= x_n + p_n\alpha_n \\ Mr_{n+1} &= Mr_n + Ap_n\alpha_n \\ s_{n+1} &= s_n + A^t(M^{-t}q_n)\alpha_n \\ & \quad \text{where } \alpha_n = (s_n^t M^{-1}(Mr_n))/((M^{-t}q_n)^t Ap_n) \\ p_{n+1} &= M^{-1}(Mr_{n+1}) - p_n\beta_n \\ M^{-t}q_{n+1} &= M^{-t}s_{n+1} - M^{-t}q_n\beta_n \\ & \quad \text{where } \beta_n = (s_{n+1}^t M^{-1}(Mr_{n+1}))/(s_n^t M^{-1}(Mr_n)) \end{aligned}$$

We see that this describes a preconditioned BiCG method on normal form, using M as a preconditioner and I as the inner product, and where $M^{-1}r_i$ are the right residuals and $M^{-t}s_i$ the left search directions. To be precise:

Lemma 82 *The left preconditioner biconjugate gradient algorithm is equivalent to a symmetrically preconditioned method with preconditioner M in that it generates the same iterates. For the residual sequence r_i generated by the left preconditioned method we find that in each iteration i, $Mr_i = Ax_i - b$.*

Proof: The rewritten algorithm is on normal form, with residuals Mr_i. The residual vectors actually computed satisfy $AM^{-1}MR = MRH$, that is, the transformed residuals Mr_i could have been derived from symmetrically preconditioned method with preconditioner M. Likewise, the left sequence S satisfies $A^tM^{-t}S = SH$. with scalars such that $S^tM^{-1}\tilde{R}$ is diagonal. •

Remark 83 *From the definition $r_1 = M^{-1}(Ax_1 - b)$ we see that r_i is related in order of magnitude to $M^{-1}b$, hence probably to $A^{-1}b$. In the context of nonlinear solves this means that we can adjust the stopping criterium to the size of the Newton update $A^{-1}f(u_i)$. This observation was made in [29].*

From the resulting equation

$$S^tM^{-1}AM^{-1}\tilde{R} = S^tM^{-1}\tilde{R}H$$

we see that the method reduces to a symmetric one for symmetric A and M.

Since $Mr_i = Ax_i - f$, we can derive the true residuals from the computed ones by applying M. This, however, may be an infeasible demand on the preconditioner; recall that normally we apply M^{-1}. Instead, we could replace the r_n-update

$$r_{n+1} = r_n + M^{-1}Ap_n\alpha_n$$

by

$$\begin{aligned} t_1 &= Ax_1 - b \\ t_{n+1} &= t_n + Ap_n\alpha_n \\ r_n &= M^{-1}t_n \end{aligned}$$

which takes one extra vector to store, but introduces no new operations. The sequence t_i then has the true residuals. Thus, we can monitor the true iterate and residual at no extra cost.

However, this monitoring presupposes that we can alter the code to add this extra vector. If the iterative method simply contains a hook for

```
monitor(A,M,xi,ri);
```

then we have incur the extra cost of computing Mr_i in order to monitor the true residual.

16.14.2 Right preconditioning

The right-preconditioned BiCG method is derived by applying an unpreconditioned algorithm to coefficient matrix AM^{-1}, right hand side b, and initial guess x_1; after the iteration stops with a final solution x_∞, the approximate solution to the original system is $M^{-1}x_\infty$.

Algorithm: Right Preconditioned BiConjugate Gradient Method

Algorithm 13 *Let A, M, x_1 and b be given. Set initially $r_1 = AM^{-1}x_1 - b$.*

Now iterate:

$$\begin{aligned} x_{n+1} &= x_n + p_n\alpha_n \\ r_{n+1} &= r_n + AM^{-1}p_n\alpha_n \\ s_{n+1} &= s_n + M^{-t}A^t q_n\alpha_n \\ & \quad \textit{where } \alpha_n = (s_n^t r_n)/(q_n^t AM^{-1}p_n) \\ p_{n+1} &= r_{n+1} - p_n\beta_n \\ q_{n+1} &= s_{n+1} - q_n\beta_n \\ & \quad \textit{where } \beta_n = (s_{n+1}^t r_{n+1})/(s_n^t r_n) \end{aligned}$$

The approximation to the true solution is $M^{-1}x_\infty$.

We rewrite, expressing the algorithm in terms of transformed quantities $M^{-t}s_n$ and $M^{-1}p_n$ as:

$$\begin{aligned} r_1 &= AM^{-1}x_1 - b \\ & \quad \alpha_n = ((M^{-t}s_n)^t M^{-1}r_n)/(q_n^t A(M^{-1}p_n)) \\ M^{-1}x_{n+1} &= M^{-1}x_n + M^{-1}p_n\alpha_n \\ r_{n+1} &= r_n + A(M^{-1}p_n)\alpha_n \\ M^{-t}s_{n+1} &= M^{-t}s_n + A^t q_n\alpha_n \\ M^{-1}p_{n+1} &= M^{-1}r_{n+1} - (M^{-1}p_n)\beta_n \\ q_{n+1} &= M^{-t}(M^{-t}s_{n+1}) - q_n\beta_n \end{aligned}$$

We see that this is a biconjugate gradient method on normal form with preconditioner M; the iterates are $M^{-1}x_i$ and the residuals are $r_i = A(M^{-1}x_i) - b$.

Lemma 84 *The right preconditioned method is equivalent to a symmetrically preconditioned method with preconditioner M and initial guess $M^{-1}x_1$ in that is generates the same residuals. For the iterates x_i generated we find that in each iteration $r_i = AM^{-1}x_i - b$.*

In other words, contrary to the left-preconditioned case, here the r_n quantities are the true residuals. However, the x_n computed are not approximations to the solution; $\tilde{x}_n = M^{-1}x_n$ are, and just for completeness we reiterate that $r_n = AM^{-1}x_n - b$, not $Ax_n - b$. Whenever needed, $\tilde{x}_n$ can be computed by a single application of M^{-1}. Thus, we can monitor the true residual at no extra cost, but monitoring the true iterates carries additional expense.

From the reduction equation

$$\tilde{S}^t M^{-1} A M^{-1} R = \tilde{S}^t M^{-1} R H$$

we see that the method again reduces in the symmetric case.

16.14.3 Two-sided preconditioning

In the two-sided method we assume that $M = M_L M_R$, and we replace the coefficient matrix by $M_L^{-1} A M_R^{-1}$. The algorithm is:

$$\begin{aligned}
r_1 &= M_L^{-1} A M_R^{-1} x_1 - M_L^{-1} b; \qquad \text{true solution is } M_R^{-1} x_\infty \\
&\quad \alpha_n = (s_n^t r_n)/(q_n^t M_L^{-1} A M_R^{-1} p_n) \\
x_{n+1} &= x_n + p_n \alpha_n \\
r_{n+1} &= r_n + M_L^{-1} A M_R^{-1} p_n \alpha_n \\
s_{n+1} &= s_n + M_R^{-t} A^t M_L^{-t} q_n \alpha_n \\
&\quad \beta_n = (s_{n+1}^t r_{n+1})/(s_n^t r_n) \\
p_{n+1} &= r_{n+1} - p_n \beta_n \\
q_{n+1} &= s_{n+1} - q_n \beta_n
\end{aligned}$$

Rewriting this in terms of $M_L r_n$, $M_R^t s_n$, $M_R^{-1} p_n$, and $M_L^{-t} q_n$:

$$\begin{aligned}
M_L r_1 &= A(M_R^{-1} x_1) - b \\
&\quad \alpha_n = \big((M_R^t s_n) M_R^{-1} M_L^{-1} (M_L r_n)\big) / \big((M_L^{-t} q_n)^t A (M_R^{-1} p_n)\big) \\
M_R^{-1} x_{n+1} &= M_R^{-1} x_n + M_R^{-1} p_n \alpha_n \\
M_L r_{n+1} &= M_L r_n + A(M_R^{-1} p_n)\alpha_n \\
M_R^t s_{n+1} &= M_R^t s_n + A^t (M_L^{-t} q_n)\alpha_n \\
M_R^{-1} p_{n+1} &= M_R^{-1} M_L^{-1} (M_L r_{n+1}) - M_R^{-1} p_n \beta_n \\
M_L^{-t} q_{n+1} &= M_L^{-t} M_R^{-t} * (M_R^t s_{n+1}) - M_L^{-t} q_n \beta_n
\end{aligned}$$

With $\tilde{R} = M_L R$ and $\tilde{S} = M_R^t S$ we find generating equations

$$\begin{aligned}
&\tilde{X} = M_R^{-1} X \text{ is a polynomial method for } A^{-1} b \\
&\quad \tilde{R} \text{ is the residual sequence } R\langle A, \tilde{X}, b\rangle \\
&\qquad\qquad AM^{-1} R = RH \\
&\qquad\qquad A^t M^{-t} \tilde{S} = \tilde{S} H
\end{aligned}$$

with $\tilde{S}^t M^{-1} \tilde{R}$ constructed to be diagonal. The reduction equation is

$$\tilde{S}^t M^{-1} A M^{-1} \tilde{R} = \tilde{S}^t M^{-1} \tilde{R} H$$

so again this reduces to the symmetric case.

Since $M_L R$ is the residual sequence of $M_R^{-1} X$, the true iterates, we can obtain the true residuals as $M_L R$, or, if this is infeasible, compute

$$\begin{aligned}
t_1 &= AM_R^{-1} x_1 - b \\
t_{n+1} &= t_n + AM_R^{-1} p_n \alpha_n \\
r_n &= M_L^{-1} t_n
\end{aligned}$$

so that $T = M_L R$.

16.14.4 The Eisenstat trick

In [25], Eisenstat gives an efficient implementation of certain preconditioned Conjugate Gradient methods, based on the following observations:

- A CG method for a system with coefficient matrix A, preconditioned by $M = LD^{-1}U \approx A$, is equivalent to one with matrix $L^{-1}AU^{-1}$ preconditioned by D^{-1}, and
- a product with $L^{-1}AU^{-1}$ can be evaluated cheaply if $A = L+U+D$ for a certain D.

We derive the first point for a Biconjugate Gradient Method:

$$\begin{aligned} x_{n+1} &= x_n + p_n\alpha_n \\ &\quad \alpha_n = \frac{s_n^t M^{-1} r_n}{q_n^t A p_n} \\ r_{n+1} &= r_n + Ap_n\alpha_n \\ s_{n+1} &= s_n + A^t q_n \alpha_n \\ p_{n+1} &= M^{-1} r_{n+1} - p_n\beta_n \\ q_{n+1} &= M^{-t} s_{n+1} - q_n\beta_n \end{aligned}$$

Rearranging the parts of M:

$$\begin{aligned} x_{n+1} &= x_n + U^{-1}(Up_n)\alpha_n \\ &\quad \alpha_n = \frac{(U^{-t}s_n)^t D(L^{-1}r_n)}{(L^t q_n)^t L^{-1}AU^{-1}(Up_n)} \\ L^{-1}r_{n+1} &= L^{-1}r_n + L^{-1}AU^{-1}(Up_n)\alpha_n \\ U^{-t}s_{n+1} &= U^{-t}s_n + (L^{-1}AU^{-1})^t(L^t q_n)\alpha_n \\ Up_{n+1} &= D(L^{-1}r_{n+1}) - Up_n\beta_n \\ L^t q_{n+1} &= D^t(U^{-t}s_{n+1}) - L^t q_n\beta_n \end{aligned}$$

which with the transformed variables

$$\tilde{r}_n = L^{-1}r_n, \quad \tilde{s}_n = U^{-t}s_n, \quad \tilde{p}_n = Up_n, \quad \tilde{q}_n = L^t q_n$$

becomes

$$\begin{aligned} \tilde{r}_1 &= L^{-1}r_1 \\ \tilde{s}_1 &= \text{arbitrary, or } U^{-t}s_1 \\ x_{n+1} &= x_n + U^{-1}\tilde{p}_n\alpha_n \\ &\quad \alpha_n = \frac{\tilde{s}_n^t D\tilde{r}_n}{\tilde{q}_n^t L^{-1}AU^{-1}\tilde{p}_n} \\ \tilde{r}_{n+1} &= \tilde{r}_n + L^{-1}AU^{-1}\tilde{p}_n\alpha_n \\ \tilde{s}_{n+1} &= \tilde{s}_n + (L^{-1}AU^{-1})^t\tilde{q}_n\alpha_n \\ \tilde{p}_{n+1} &= D\tilde{r}_{n+1} - \tilde{p}_n\beta_n \\ \tilde{q}_{n+1} &= D^t\tilde{s}_{n+1} - \tilde{q}_n\beta_n \end{aligned}$$

The second observation is that, with $A = L+U+K$ where $K = D_A - D_L - D_U$,

$$L^{-1}AU^{-1} = L^{-1}(L+U+K)U^{-1} = U^{-1} + L^{-1}(I+KU^{-1}).$$

In other words, applying A and solving with M can be done for a joint cost of one upper and one lower solve, multiplication by K, and two vector additions.

Given the requirements that $M = LD^{-1}U$ and $A = L+U+K$, it is most natural to write $M = (L_A+D)D^{-1}(D+U_A)$. Thus we could construct D by a point factorisation (including SSOR) of A, but everything generalises to block versions of these methods. Multiplication by $K = D_A - 2D$ is straightforward, in both the point and the block case.

16.15 Preconditioning by the symmetric part and separable approximations

As symmetric matrices are often more pleasant to work with than non-symmetric ones, one could consider preconditioning a matrix A with its symmetric part $Q = (A + A^t)/2$. In a generalization of this idea, one could precondition by an approximation of the symmetric part; a separable one is a particularly good idea, since those can be solved very efficiently.

Here are some of the forms this idea has gone through.

- Solving a linear system from a **separable** equation

  $$Au = -(a(x)u_x)_x - (b(y)u_y)_y$$

  can be done by *fast solvers* such as the Fast Fourier Transform in $O(N \log N)$ operations or thereabouts [73].
- If A is **nonseparable, but self-adjoint elliptic,** approximating it with a *self-adjoint separable* operator M gives a spectrally equivalent preconditioner. D'Yakonov [23] considered ADI as a preconditioner; Widlund [84] used a stationary iteration method here, which Concus and Golub [15] accelerated with Chebyshev iteration, and Bank [6] (using the marching algorithm) by Conjugate Gradients.
- The general, **nonself-adjoint, nonseparable, elliptic** case was discussed by Concus and Golub [16] and Widlund [85]. Elman and Schultz [27] proved spectral equivalence for self-adjoint preconditioners, but gave numerical evidence that nonself-adjoint preconditioners give similar asymptotic rates.

16.16 Diagonal coefficient matrices

While it is silly to solve a system with a coefficient matrix of diagonal form by an iterative method, such matrices can be good test problems. Stationary iterative methods converge in one step with such a matrix, but, surprisingly, conjugacy based methods do not see the form of the matrix, other than through different roundoff behaviour.

Question: *Prove this.*

16.17 CG in Function Spaces

Conjugate Gradients in Hilbert Space

On April 2, 2003 I asked in na.digest@na-net.ornl.gov
the question:

> Is anything known about the convergence of the conjugate gradient method for linear equations in function spaces?
>
> E.g., is it convergent for a linear system Ax=b if A is self-adjoint, positive definite, and 1-A is compact? What about the rate of convergence?

Here is a summary of the responses that I got.
If you have more precise information at this level of brevity, please inform me at Arnold.Neumaier@univie.ac.at

General comments:

The earliest paper is the thesis of Hayes in 1954.
Zuhair Nashed and colleagues have written extensively about cg on function spaces. The earlier references can be found in [Golub and O'Leary].

CG can be applied to any linear variational problem of the form
  Find u in V so that:
  a(u,v)=L(v), for all v in V,
where:
V is a real Hilbert space with norm ||.||,
a:VxV--->R is bilinear, continuous, symmetric with
a(v,v) \ge c||v||^2 for all v in V, with c>0, and
L:V--->R is linear and continuous. [see Glowinski]

The proof of the "standard" convergence rate estimate goes as for matrices, but one needs to use the spectral theory for self-adjoint operators in a Hilbert space. Compactness is not ne

If I-A is compact, the convergence is superlinear, due to the clustering of eigenvalues at 1. [see Widlund, or the book by Nev [Axelsson and Karatson] distinguish a sublinear, a linear, and a superlinear phase.

[Kelley, Campbell, Ipsen, Meyer] give a nice proof of the superl convergence of GMRES for I-A compact. The superlinear convergenc of CG then follows from a general version of the Brown result fo orthogonal/minimal residual method pairs:
 ||r^{OR}_k|| \le ||r^{MR}_k||/sqrt{1-||r^{MR}_k||^2/||r^{MR}_{k
where r^{OR}_k and r^{MR}_k are the kth residuals of the orthogonal and minimal residual methods, respectively.

Essentially the same results hold for GMRES and several quasi-Newton methods.

If A is compact, the solution of Ax=b is ill-posed, but CG has a regularizing effect; it converges (for selfadjoint semidefinite compact or not) if a solution exists to the minimal norm solutio [see Hanke]. But in the presence of discretization errors or rou errors, one may (in the compact case) not iterate for too long; otherwise the information gained is destroyed again.

References (in chronological order)

Hayes, R.M.,
Iterative methods of solving linear problems on Hilbert space,
in: Contributions to the Solution of Systems of Linear Equations
and the Determination of Eigenvalues (O. Taussky, ed.),
National Bureau of Standard, Appl. Math. Ser., 39 (1954), 71-103.

J.W. Daniel,
The conjugate gradient method for linear and nonlinear operator
equations,
SIAM J. Numer. Anal. 4 (1967), 10-26.

J.W. Daniel,
The Approximate Minimization of Functionals,
Prentice-Hall, Englewood Cliffs, N.J., 1971.

W.J. Kammerer and M.Z. Nashed,
On the convergence of the conjugate method for singular linear
operator equations in Hilbert spaces,
SIAM J. Numerical Analysis 9 (1972), 165-181.

Paterson, W.M.,
Iterative methods for the solution of a linear operator
equation in Hilbert space - a survey,
Lecture Notes in Mathematics 394,
Springer Verlag (1974?).

R. Winther,
A numerical Galerkin method for a parabolic problem,
Ph.D. Thesis, Dept. of Computer Science, Cornell University,
Ithaca, New York 1977.

O. Widlund
SIAM J. Num. Anal. 15 (1978), 801-812.

Fischna, Z.,
Some convergence properties of the conjugate gradient method in
Hilbert space,
SIAM J. Numer. Anal., 16 (1979), 380-384.

R. Winther,
Some superlinear convergence results for the conjugate gradient
SIAM J. Numer. Anal. 17 (1980), 14-17.
[Math. Rev. 81k:65060]

C.T. Kelley and E.W. Sachs,
Quasi-Newton methods and unconstrained optimal control problems,
SIAM J. Contr. Opt 25 (1987), 1503-1517.

G.H. Golub and D.P. O'Leary,
Some history of the conjugate gradient and Lanczos algorithms:
1948-1976,
SIAM Review 31 (1989), 50-102.

T. Kerkhoven and Y. Saad,
On acceleration methods for coupled nonlinear elliptic systems,
Numer. Math. 60 (1992), 525-548.

D.M. Hwang and C.T. Kelley,

Convergence of Broyden's method in Banach spaces,
SIAM J. Opt. 2 (1992), 505-532.

O. Nevanlinna,
Convergence of Iterations for Linear Equations,
Birkha"user Verlag, Basel, 1993

S.L. Campbell, I.C.F. Ipsen, C.T. Kelley and C.D. Meyer,
GMRES and the minimal polynomial,
BIT 36 (1996), 664-675.

S.L. Campbell, I.C.F. Ipsen, C.T. Kelley, C.D. Meyer and Z.Q. Xue,
Convergence estimates for solution of integral equations with GMRES,
J. Integral Eqs. and Applications 8 (1996), 19--34.

K. Atkinson,
The Numerical Solution of Integral Equations of the Second Kind,
Cambridge University Press, 1997.
[I-A compact: Section 4.6]

K. Atkinson and Weimin Han,
Theoretical Numerical Analysis: A Functional Analysis Framework,
Springer-Verlag, 2001.
[Section 4.6]

O. Axelsson and J. Karatson,
On the rate of convergence of the conjugate gradient method for
linear operators in Hilbert space,
Numer. Funct. Anal. 23 (2002), 285-302.

R. Glowinski,
Finite element methods for incompressible viscous flow,
Chapt. III in:
Handbook of Numerical Analysis, Volume IX,
North-Holland, 2003.

The ill-posed case

The following references deal with Ax=b, where A is compact
(using regularization).

J. Baumeister,
Stable Solution of Inverse Problems,
Vieweg Verlag, Braunschweig 1987.

H. Brakhage,
On ill-posed problems and the method of conjugate gradients,
pp. 165-175 in:
Inverse and Ill-Posed Problems
(H.W. Engl and C.W. Groetsch, eds.),
Academic Press, Boston 1987.

A.K. Louis,
Convergence of the conjugate gradient method for compact operato
pp. 177-183 in:
Inverse and Ill-Posed Problems
(H.W. Engl and C.W. Groetsch, eds.),
Academic Press, Boston 1987.

A.K. Louis,
Inverse und schlecht gestellte Probleme,
Teubner Verlag, Stuttgart 1989.

Martin Hanke,
Conjugate Gradient Type Methods for Ill-Posed Problems,
Pitman Research Notes in Mathematics,
Longman House, Harlow, Essex, 1995.

17 History

Previous section: 16 Next section: **??**

The evolution of the conjugate gradient method and other Krylov methods has proceeded along branching paths. Here are some of the defining papers.

Lanczos 1950 [47] Cornelius Lanczos proposed three-term recurrences $b_{n+1} = Ab_{n-1} - \alpha_n b_n - \beta_{n-1} b_{n-1}$ for the solution of the eigenvalue problem. Orthogonality of the b_n vectors follows from the requirement that b_{n+1} be of minimal length (Lanczos credits the idea of successive orthogonalisation of a sequence to [59]). The generalisation to a nonsymmetric matrix is by generating two mutually orthogonal sequences, the second generated from $b^*_{n+1} = A^t b^*_{n-1} - \alpha_n b^*_n - \beta_{n-1} b^*_{n-1}$, that is, both satisfying a three-term recurrence. No normalisation of the sequences is applied, with as result $b_n = P_n(A)b_1$ where the polynomial P_n is normalised to $P_n(x) = x^n + \cdots$. Although the sequences satisfy three-term recurrences, the actual algorithm uses coupled two-term recurrences.

Arnoldi 1951 [2] W.E. Arnoldi, also targeting the eigenvalue problem, gave a further discussion of the Lanczos method, and proposed a variant, which he terms a Galerkin method, leaving out the A^t-based sequence b^*n. As a result he derives an upper Hessenberg matrix of $b^t_i A b_j$ coefficients.

Lanczos 1952 [48]
Hestenes and Stiefel 1952 [41]
Reid 1971 [65]
Fletcher 1975 [32]
Axelsson 1980 [4]
O'Leary 1980 [58]
Young and Jea 1980 [44]
Faber and Manteuffel 1984 [31]

Sonneveld 1989 [72]
van der Vorst 1992 [76]

For an extensive bibliography we refer to [37].

References

[1] Mario Arioli, Iain Duff, and Daniel Ruiz. Stopping criteria for iterative solvers. *SIAM J. Matrix Anal.*, 13:138–144, 1992.

[2] W.E. Arnoldi. The principle of minimzed iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[3] Steven F. Ashby. Some paper about stopping tests. Technical report, Lawrence Livermore, I guess.

[4] Owe Axelsson. Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations. *Lin. Alg. Appl.*, 29:1–16, 1980.

[5] Owe Axelsson and Gunhild Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numer. Math.*, 48:499–523, 1986.

[6] Randolph E. Bank. Marching algorithms for elliptic boundary value problems; II: the variable coefficient case. *SIAM J. Numer. Anal.*, 14:950–970, 1977.

[7] Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia PA, 1994. `http://www.netlib.org/templates/`.

[8] Christian H. Bischof. Incremental condition estimation. *SIAM J. Matrix Anal. Appl.*, 11:312–322, 1990.

[9] A. Björck. Numerics of Ggram-Schmidt orthogonalization. *Lin. Alg. Appl.*, pages 297–316, 1994.

[10] A. Björck and C.C. Paige. Loss and recapture of orthogonality in the modified Ggram-Schmidt algorithm. *SIAM J. Mat. Anal. Appl.*, pages 176–190, 1992.

[11] S.L. Campbell, I.C.F. Ipsen, C.T. Kelly, and C.D. Meyer. GMRES and the minimal polynomial. 36:664–675, 1996.

[12] T. Chan and W. Wan. Analysis of project methods for solving linear

systems with multiple right-hand sides. *SIAM J. Sci. Comput.* also UCLA CAM report 94-26; to appear.

[13] Tony F. Chang and Michael K. Ng. Galerken projection methods for solving multiple linear systems. Technical Report CAM Report 96-31, UCLA Department of Mathematics, Computational and Applied Mathematics, 1996.

[14] A. Chronopoulos and C.W. Gear. *s*-step iterative methods for symmetric linear systems. *Journal of Computational and Applied Mathematics*, 25:153–168, 1989.

[15] Paul Concus and Gene H. Golub. Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations. *SIAM J. Numer. Anal.*, 10:1103–1120, 1973.

[16] Paul Concus and Gene H. Golub. A generalized conjugate gradient method for nonsymmetric systems of linear equations. In *Compute methods in applied sciences and engineering, second international symposium, Dec 15–19, 1975; Lecture Notes in Economics and Mathematical Systems, vol 134*, Berlin, 1976. Spriner Verlag. Stanford University technical report STAN-CS-76-535.

[17] J. Cullum and A. Greenbaum. Relations between Galerkin and norm-minimizing iterative methods for solving linear systems. *SIAM J. Matrix Anal.*, 17:223–247, 1996.

[18] J.W. Daniel, W.B. Gragg, L. Kaufman, and G.W. Steward. Reorthogonaliation and stable algorithms for updating the Gram-Schmidt QR factorization. *Mathematics of Computation*, 30:772–795, 1976.

[19] E.F. D'Azevedo, V.L. Eijkhout, and C.H. Romine. Lapack working note 56: Reducing communication costs in the conjugate gradient algorithm on distributed memory multiprocessor. Technical Report CS-93-185, Computer Science Department, University of Tennessee, Knoxville, 1993.

[20] E.F. D'Azevedo and C.H. Romine. Reducing communcation costs in the conjugate gradient algorithm on distributed memory multiprocessors. Technical Report ORNL/TM-12192, Oak Ridge National Lab, 1992.

[21] E. de Sturler. A parallel restructured version of GMRES(m). Technical Report 91-85, Delft University of Technology, Faculty of Technical Mathematics and Informatics, 1991.

[22] J. Demmel, M. Heath, and H. Van der Vorst. Parallel numerical linear algebra. In *Acta Numerica 1993*. Cambridge University Press, Cambridge, 1993.

[23] E.G. D'Yakonov. The method of variable directions in solving systems of finite difference equations. *Soviet Mathematics / Doklady*, 2:577–580, 1961. TOM 138, 271–274.

[24] Victor Eijkhout. Lapack working note 51: Qualitative properties of the conjugate gradient and lanczos methods in a matrix framework. Technical Report CS 92-170, Computer Science Department, University of Tennessee, 1992.

[25] Stanley C. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM J. Sci. Stat. Comput.*, 2:1–4, 1981.

[26] Howard C. Elman. *Iterative Methods for Large Sparse Nonsymmetric Systems of Equations.* PhD thesis, 1982.

[27] Howard C. Elman and Martin H. Schultz. Preconditioning by fast direct methods for non self-adjoint nonseparable elliptic equations. *SIAM J. Numer. Anal.*, 23:44–57, 1986.

[28] M. Engeli, M. Ginsburg, H. Rutishauser, and E. Stiefel. Refined iterative methods for the computation of the solution and the eigenvalues of self-adjoint boundary value problems. Technical Report 8, Mitt. Inst. Angew. Math. ETH Zürich, 1959.

[29] Alexandre Ern, Vincent Giovangigle, David E. Keyes, and Mitchell D. Smooke. Towards polyalgorithmic linear system solvers for nonlinear elliptic problems. *SIAM J. Sci. Comput.*, 15:681–703, 1994.

[30] M. A. Krasnoselskii et al. *Approximate Solution of Operator Equations*. Wolters-Noordhoff, Groningen, 1972.

[31] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21:352–362, 1984.

[32] R. Fletcher. Conjugate gradient methods for indefinite systems. In G.A. Watson, editor, *Numerical Analysis Dundee 1975*, pages 73–89, New York, 1976. Springer Verlag.

[33] Roland W. Freund. Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Stat. Comput*, 13:425–448, 1992.

[34] Roland W. Freund. A tranpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14(2):470–482, 1993.

[35] Roland W. Freund and Noël M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[36] Gene H. Golub, Franklin T. Luk, and Michael L. Overton. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Trans. Math. Software*, 7:149–169, 1981.

[37] Gene H. Golub and Dianne P. O'Leary. Some history of the conjugate gradient and Lanczos methods. *SIAM Review*, 31:50–102, 1989.

[38] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, second edition edition, 1989.

[39] Louis A. Hageman and David M. Young. *Applied Iterative Methods*. Academic Press, New York, 1981.

[40] K. Hessenberg. *Auflösung linearere Eigenweraufgaben mit Hilfe der Hamilton-Cayleyschen Gleichung, Dissertation*. Technische Hochschule, Darmstadt, 1941.

[41] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Nat. Bur. Stand. J. Res.*, 49:409–436, 1952.

[42] Alston S. Householder. *The theory of matrices in numerical analysis*. Blaisdell Publishing Company, New York, 1964. republished by Dover Publications, New York, 1975.

[43] W. Jalby and B. Philippe. Stability analysis and improvement of the block Gram-Schmidt algorithm. *SIAM J. Sci. Stat. Comput.*, 5:1058–1073, 1991.

[44] Kang C. Jea and David M. Young. Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods. *Lin. Alg. Appl.*, 34:159–194, 1980.

[45] E.F. Kaasschieter. A practical termination criterion for the conjugate gradient method. *BIT*, 28:308–322, 1988.

[46] D. K. Kaushik, D. E. Keyes, and B. F. Smith. On the interaction of architecture and algorithm in the domain-based parallelization of an unstructured grid incompressible flow code. In *Proceedings of the 10th Intl. Conf. on Domain Decomposition Methods, J. Mandel et al., eds.*, pages 311–319, 1998.

[47] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research, Nat. Bu. Stand.*, 45:255–282, 1950. Research Paper 2133.

[48] C. Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Research, Nat. Bu. Stand.*, 49:33–53, 1952.

[49] C. Lanczos. *Linear Differential Operators*. Van Rostrand, London, 1961.

[50] John Gregg Lewis and Ronald G. Rehm. The numerical solution of a nonseperable elliptic partial differential equation by preconditioned conjugate gradients. *J. Res. Nat. Bureau of Standards*, 85:367–390, 1980.

[51] D.G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, Mass., 1965.

[52] T.A. Manteuffel. The Tchebychev iteration for nonsymmetric linear systems. *Numer. Math.*, 28:307–327, 1977.

[53] T.A. Manteuffel. Adaptive procedure for estimating parameters of the nonsymmetric Tchebychev iteration. *Numer. Math.*, 31:183–208, 1978.

[54] Gerard Meurant. Multitasking the conjugate gradient method on the CRAY X-MP/48. *Parallel Computing*, 5:267–280, 1987.

[55] R.B. Morgan. Computing interior eigenvalues of large matrices. *Lin. Alg. Appl.*, 154/156:289–309, 1991.

[56] R.A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM J. Numer. Anal.*, 24, 1987.

[57] W. Oettli and W. Prager. Compatibility of aproximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numer. Math.*, 6:405–409, 1964.

[58] Dianne P. O'Leary. The block conjugate gradient algorithm and related methods. *Lin. Alg. Appl.*, 29:293–322, 1980.

[59] O.Száz. *Math. és phys. lapok*, 19:221–227.

[60] C.C. Paige. Computational variants of the Lanczos method for the eigenproblem. *J. Inst. Maths Applics*, 10:373–381, 1972.

[61] C.C. Paige. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *J. Inst. Maths Applics*, 18:341–349, 1976.

[62] C.C. Paige, B.N. Parlett, and H.A. van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. 2:115–134, 1995.

[63] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[64] Christopher C. Paige and Michael A. Saunders. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8:43–71, 1982.

[65] J.K. Reid. On the method of conjugate gradients for the solution of large sparse systems of linear equations. In J.K. Reid, editor, *Large sparse sets of linear equations*, pages 231–254. Academic Press, London, 1971.

[66] J.L. Rigal and J. Gaches. On the compatibility of a given solutin with the data of a linear system. *J. Assoc. Comput. Mach.*, 14:543–548, 1967.

[67] Yousef Saad. Practical use of some krylov subspace methods for solving indefinite and nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 5:203–228, 1984.

[68] Yousef Saad and Martin H. Schultz. Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Math. Comp.*, 44:417–424, 1985.

[69] Yousef Saad and Martin H. Schultz. GMRes: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[70] W. Schoenauer. *Scientific Computing on Vector Computers*. North-Holland, 1987.

[71] C. Smith, A. Peterson, and R. Mittra. A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields. *IEEE Transactions on Antennae and Propagation*, 37:1490–1493, 1989.

[72] Peter Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.

[73] P.N. Swarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Review*, 19:490–501, 1977.

[74] A. van der Sluis and H.A. van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48:543–560, 1986.

[75] H.A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Num. Lin. Alg. with Appls.*, 1:1–7, 1993.

[76] Henk van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.

[77] Henk A. van der Vorst and Tony F. Chan. Linear system solvers:

Sparse iterative methods. Technical Report CAM Report 94-28, Department of Mathematics, University of California, Los Angeles, 1994.

[78] J. Van Rosendale. Minimizing inner product data dependencies in conjugate gradient iteration. Technical Report 172178, ICASE, NASA Langley Research Center, Hampton, Virginia, 1983.

[79] Denis Vanderstraeten. A stable and efficient parallel block gram-schmidt algorithm. In P. Amestoy et al, editor, *Proceedings of the Euro-Par '99 Conference*, pages 1128–1135, Toulouse, 1999.

[80] P.K.W. Vinsome. ORTHOMIN, an iterative method for solving sparse sets of simultaneous linear equations, paper SPE 5729. In *4th Symposium of Numerical Simulation of Reservoir Performance of the Society of Petroleum Engineers of the AIME, Los Angeles, 19–20 February 1976.*

[81] V.V. Voevodin. The problem of non-self-adjoint generalization of the conjugate gradient method is closed. *USSR Comput. Maths. Math. Phys.*, 23:143–144, 1983.

[82] H.F. Walker. Implementation of the GMRES method using Householder transformations. *SIAM J. Sci. Stat. Comput.*, 9:152–163, 1988.

[83] R. Weiss. *Convergence Behavior of Generalized Conjugate Gradient Methods.* PhD thesis, University of Karlsruhe, 1990.

[84] O. Widlund. On the use of fast methods for separable finite difference equations for the solution of general elliptic problems. In D.J. Rose and R.A. Willoughby, editors, *Sparse matrices and their applications*, pages 121–134. Plenum Press, New York, 1972.

[85] O. Widlund. A Lanczos method for a class of non-symmetric systems of linear equations. *SIAM J. Numer. Anal.*, 15:801–812, 1978.

[86] J. Wilkinson. *The Algebraic Eigenvalue Problem.*

[87] Lu Zhou and Homer F. Walker. Residual smoothing techniques for iterative methods. *SIAM J. Sci. Stat. Comput.*, 15:297–312, 1992.

A Matlab code examples

Previous section: **??** Next section: 2

A.1 Minimum residual methods

A.1.1 GMRES

A.2 Polynomial squaring methods

A.2.1 Conjugate Gradient Squared

A.3 Arnoldi method

See section A.5.1 for the `xqr` routine.

A.4 Householder reflectors

These code examples illustrate the theory of section 11.4.

A.4.1 Basic reduction routines

Reduction to Hessenberg form by Householder reflectors:

A.4.2 Retrieving Householder vectors from a Lanczos basis

A.5 Utilities

A.5.1 Orthogonalisation

Index

affine combinations, 20
Axelsson's method, 64

backward error, 98
BLAS
    in search direction update, 51

consistency, 106

FOM, 48
forward error, 98

GCR, 64
GMRES, 63–64
GMRESr, 65
Gram-Schmidt, 63, 80–81

Left preconditioning
    in nonlinear solvers, 123
line search, 35

Marching algorithm, 127
MINRES, 62
moments, 79

Newton method
    and left preconditioning, 123
null space, 106
nullspace, 106

OrthoDir, 64
OrthoMin, 64
OrthoRes, 48

Power method, 18
Preconditioner, 11
Projection, 43–44

QMR, 66

Residual smoothing, 66–69
Ritz values, 101
    harmonic, 101
Ritz vectors, 101

Search directions, 57–58
seed systems, 35
Singular values, 114