National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

McGill University
School of Computer Science

Matrix Based Derivations and Representations of Krylov Subspace Methods

Penelope L. Anderson

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Master of Science

ISBN 0-612-12154-2

Acknowledgements

I would like to thank Professor Chris Paige for his guidance, support, patience and encouragement. My office-mate Xiaowen Chang for his help, friendship and the occasional dim sum. My family for their love and support whether near or far. My friends and room-mates for putting up with me throughout this process.

This thesis is dedicated to my step-father, John L. Hastings.

Abstract

This thesis is based on recent work by Paige which gave a formalism for presenting and analyzing the class of algorithms which manipulate an appropriate Krylov subspace in solving large sparse systems of linear equations. This formalism — a way of dividing a method of solution into a Krylov process and an associated subproblem — is described and then applied to several of the more popular algorithms in use today including the methods of Conjugate Gradients and Bi-Conjugate Gradients. The aim is to clarify these algorithms to make them easier to understand, analyze and use. Several of the methods presented in this thesis were developed in exactly this way — notably the Symmetric LQ method and the Generalized Minimum Residual method — and required little or no effort to characterize using the formalism. It was successfully applied to Conjugate Gradients and BiConjugate Gradients, already recognized as being closely related to the symmetric and unsymmetric Lanczos processes respectively. The newer algorithms such as Conjugate Gradients Squared and BiConjugate Gradients Stabilized, with less obvious relation to a specific Krylov process, provided more difficulty in their clarification.

Résumé

Ce mémoire est basé sur le travail récent de Paige sur un formalisme pour présenter et analyser une classe d'algorithmes manipulant un sous-espace de Krylov approprié dans la résolution de larges systèmes creux d'équations linéaires. Ce formalisme — une façon de diviser une méthode de résolution en un processus de Krylov et un sous-problème associé — est décrit et ensuite appliqué à plusieurs des algorithmes les plus populaires en ce moment, dont la méthode des Gradients Conjugués et celle des Gradients BiConjugués. Le but de ce mémoire est de clarifier ces algorithmes pour les rendre plus faciles à comprendre, analyser et utiliser. Plusieurs des méthodes ici présentées ont justement été développées de cette façon — la méthode des LQ Symétriques et la méthode des Résidus Minimums Généralisés en particulier — et leur caractérisation à l'aide du formalisme n'a requis que peu d'efforts. Le formalisme a été appliqué aux Gradients Conjugués et aux Gradients BiConjugués, connus pour être liés respectivement aux processus symétriques et asymétriques de Lanczos. Les algorithmes plus récents, tel que les Carrés des Gradients Conjugués et les Gradients BiConjugués Stabilisés, dont le lien avec un processus de Krylov spécifique est moins évidente, ont été plus difficiles à clarifier.

Contents

1 Introduction

1.1 Formalism

Based on ideas in [PS75], Paige [Pai94] states a simple formalism for motivating, presenting and studying Krylov subspace methods for problems involving a matrix A. These methods are useful for example in solving the eigenproblem $Ax = \lambda x$ or solution of equations $Ax = b$ when A is large and sparse.

Given a matrix $A \in R^{n \times n}$, and a starting vector v, a *Krylov Process* produces vectors $\{v_1, v_2, \ldots, v_k\}$ which span the *k-th Krylov subspace*

$$\mathcal{K}^k(A; v) \equiv \text{span}\{v, Av, \ldots, A^{k-1}v\}.$$

Krylov subspace methods seek elements of these subspaces which are in some sense good approximations to the solutions of the original problems.

The formalism for Krylov subspace methods is as follows:

A Krylov subspace process + Solving a subsidiary problem
→ A Krylov subspace method.

The terms used above were chosen carefully to distinguish the parts of the algorithm.

Process refers to the particular process we are considering, for example, the Lanczos process, i.e. the way in which the vectors are formed to span the Krylov subspace in question. Many processes are presented in § 2.

Method refers to the theoretical method used to solve the problem. For example [BBC+94] gives one implementation of the Conjugate Gradients, or CG method for solving $Ax = b$; another implementation using the Lanczos process and then considering iterates $x_k \in \mathcal{K}^k(A; b)$ obtained by manipulating the Lanczos vectors is well known, see for example [GL89, § 9.3.1] and

[PS75] and § 4.1 in this thesis. Although the implementations are different, the resulting methods are theoretically the same (using exact arithmetic).

For any underlying Krylov process, the manipulation of the vectors obtained in that process is known as the *subproblem* for this process used to obtain this method.

A significant advantage of this approach is that both the *process* and the *subsidiary problem* may be presented as clear matrix formulations, and these make the *method* easy to understand and analyse. The less transparent vector representations (i.e. the implementations) of the method can then be obtained from these matrix formulations. The vector representation is important for implementation, but the matrix representation is a far more powerful tool for teaching, for understanding, for relating and deriving different methods, and for analysing their properties.

Paige and Saunders [PS75] implicitly used this formalism to understand old methods, and to develop new methods. However, several of the more recent methods in this area appear to be significantly more complicated than those in [PS75], and not nearly as well motivated. The result is that many users of these algorithms have little understanding of them, a fact which does not help either their use or implementation.

In this thesis the formalism is applied to the more successful Krylov subspace methods for $Ax = b$ used today – for example GMRES (Generalized Minimum Residual), BiCG (BiConjugate Gradients), QMR (Quasi-Minimal Residual), CGS (Conjugate Gradients Squared) – all of which are summarized, usually in vector form, in [BBC+94].

The aim is to see if the formalism described here, which was so successful in clarifying earlier methods, can also help to simplify the understanding and development of today's more complicated methods.

1.2 Notation

- Matrices are represented by upper case Roman letters such as A or B_k.

- Vectors are represented by lower case Roman letters such as v_k, including the columns of matrices, for example w_j is the j-th column of W_k. Usually columns of a matrix use the same letter as the matrix itself; one prominent exception is e_j which denotes the j-th column of a unit matrix I of appropriate dimension. Occasionally, for clarity, superscripts such as the k in $e_i^{(k)}$ will be used to specify the dimension.

- Lower case Greek letters represent scalars. The scalar entries of matrices or vectors may be represented by lower case Greek letters or by lower case Roman letters the same as the entity itself, with appropriate subscripts. For example, the entries of the vector z_k are ζ_j; the entries of the matrix H_k are h_{ij}.

- c_k and s_k represent $cos(\theta_k)$ and $sin(\theta_k)$ respectively in those algorithms which involve a QR or LQ decomposition.

- $\|\cdot\|$ is used throughout to denote the 2-norm.

1.3 Table of Abbreviations

QR	referring to the decomposition of a matrix into an orthogonal matrix and an upper triangular matrix
LQ	referring to the decomposition of a matrix into a lower triangular matrix and an orthogonal matrix
CG	Conjugate Gradients
SYMMLQ	Symmetric LQ
MINRES	Minimum Residual
CGNE	Conjugate Gradients on the Normal Equations
LSQR	Least Squares with the QR decomposition
GMRES	Generalized Minimum Residual
BiCG	BiConjugate Gradients
QMR	Quasi-Minimal Residual
CGS	Conjugate Gradients Squared
BiCGSTAB	BiConjugate Gradients Stabilized
SPD	Symmetric Positive Definite

1.4 A note on the literature review

This thesis is required to have a review of the literature. In that the body of this thesis describes popular methods of solution of large sparse systems of linear equations and clearly refers to the original papers and subsequent articles written on these methods, it is our belief that this review is implicit and need not be presented separately.

2 Krylov Processes

2.1 The general Krylov process

We first state a very general Krylov process, then derive some of the more popular processes from this by manipulating its matrix formulation. We feel this is briefer and clearer than the more usual approaches to deriving these processes.

The general Krylov process with a given matrix $A \in \mathcal{R}^{n\times n}$ and starting vector $v \in \mathcal{R}^n$ forms a sequence of vectors $v_1, v_2, \ldots \in \mathcal{R}^n$ as follows

$$\begin{aligned} h_{1,0}v_1 &= v \\ h_{j+1,j}v_{j+1} &= Av_j - h_{j,j}v_j - \cdots - h_{1,j}v_1, \quad j = 1,2,\ldots, \end{aligned} \tag{1}$$

where if $h_{j+1,j}v_{j+1} = 0$ the process is stopped. Later the coefficients $h_{i,j}$ will be chosen to give different processes. Clearly after $k-1$ steps $\{v_j\}_1^k$ spans the k-th Krylov subspace

$$\mathcal{K}^k(A;v) \equiv \text{span}\{v, Av, \ldots, A^{k-1}v\}. \tag{2}$$

After k steps (1) corresponds to the columns of

$$AV_k = V_kH_k + h_{k+1,k}v_{k+1}e_k^T = V_{k+1}H_{k+1,k} \tag{3}$$

$$V_k \equiv \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix},$$

$$H_k \equiv \begin{bmatrix} h_{1,1} & h_{1,2} & \cdot & \cdot & h_{1,k} \\ h_{2,1} & h_{2,2} & \cdot & \cdot & h_{2,k} \\ & \cdot & \cdot & \cdot & \cdot \\ & & \cdot & \cdot & \cdot \\ & & & h_{k,k-1} & h_{k,k} \end{bmatrix}, \quad H_{k+1,k} \equiv \begin{bmatrix} H_k \\ h_{k+1,k}e_k^T \end{bmatrix}.$$

Notice that $H_{k+1,k}$ is just the upper Hessenberg H_k supplemented by a row of zeros except for the last element. If $h_{k+1,k}v_{k+1} = 0$ then $AV_k = V_kH_k$, and

$\mathcal{K}^k(A;v)$ is an invariant subspace of A, which we usually want. However the reverse need not hold for the general process, so the specific Krylov processes we consider will (in theory, that is in the absence of rounding errors) ensure termination, i.e. $h_{k+1,k}v_{k+1} = 0$, immediately the columns of V_k span an invariant subspace of A.

Householder [Hou74] gives a detailed treatment of Krylov's method [Kry31] and its relation to other algorithms for the algebraic eigenvalue problem.

2.2 The Arnoldi algorithm

One obvious choice to ensure termination is to derive orthogonal vectors $v_1, v_2, \ldots$ in (1). Here and later we will use simple inductive arguments to obtain expressions for coefficients. Suppose we have nonsingular

$$V_k^T V_k = D_k = \mathrm{diag}(\rho_1, \ldots, \rho_k), \tag{4}$$

which is trivially obtained for $k = 1$, then from (3)

$$V_k^T A V_k = D_k H_k + h_{k+1,k} V_k^T v_{k+1} e_k^T,$$

and to ensure $V_k^T v_{k+1} = 0$ in theory we take

$$h_{i,k} = v_i^T A v_k / \rho_i, \quad i = 1, \ldots, k. \tag{5}$$

If we were to compute the vectors via (5) and then (1), this would correspond to the numerically inadequate classical Gram-Schmidt approach, see for example [GLS9, § 5.2], so a more numerically reliable computation is via a modified Gram-Schmidt approach

$$\begin{aligned} u_k &= A v_k \\ \text{for } i &= 1, \ldots, k \end{aligned}$$

$$h_{i,k} = v_i^T u_k / \rho_i$$

$$u_k = u_k - h_{i,k} v_i$$

$$\text{and then} \quad h_{k+1,k} v_{k+1} = u_k.$$

This gives (3) with

$$V_k^T V_k = D_k, \quad V_k^T v_{k+1} = 0, \quad V_k^T A V_k = D_k H_k. \tag{6}$$

If $h_{k+1,k} v_{k+1}$ is then nonzero we take $h_{k+1,k}$ to be whatever we want. For orthonormal vectors we choose $h_{k+1,k}$ to give $\|v_{k+1}\| = 1$, and we can continue the process. So in this case (5) and (6) become

$$h_{i,k} = v_i^T A v_k, \quad i = 1, \ldots k, \tag{7}$$

$$V_k^T V_k = I, \quad V_k^T v_{k+1} = 0, \quad V_k^T A V_k = H_k. \tag{8}$$

This was proposed by Arnoldi for solving the eigenproblem: on termination all the eigenvalues of H_k are eigenvalues of A, see [Arn51].

2.3 The Lanczos process for symmetric A

If A is symmetric in the Arnoldi algorithm with arbitrary $h_{j+1,j}$ leading to (6), we see $V_k^T A V_k = D_k H_k = H_k^T D_k$ is tridiagonal and symmetric. That means $h_{i,j} = 0, \ \ i = 1, 2, \ldots, j-2$ in (1), and from the $j-1, j$ element, see (4),

$$\rho_{j-1} h_{j-1,j} = \rho_j h_{j,j-1}. \tag{9}$$

Let us define $\alpha_j \equiv h_{j,j}, \ \ \beta_j \equiv h_{j,j-1}, \ \ \gamma_j \equiv h_{j-1,j}$. This leads to Lanczos' 3-term process with arbitrary normalization

$$\gamma_0 v_0 \equiv 0$$

$$\beta_1 v_1 = v, \quad \beta_1 \neq 0 \ \text{ arbitrary},$$

$$\text{for} \quad j = 1, \ldots, k$$

$$\alpha_j = v_j^T A v_j/\rho_j$$

$$\gamma_j = \beta_j \rho_j/\rho_{j-1} \tag{10}$$

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \gamma_j v_{j-1}, \tag{11}$$

$$\beta_{j+1} \neq 0 \quad \text{arbitrary}$$

$$\text{stop if} \quad \beta_{j+1} v_{j+1} = 0.$$

Notice the definition (10) of γ_j agrees with the original (5) both from symmetry (9) which says

$$\rho_{j-1}\gamma_j = \rho_j \beta_j, \tag{12}$$

or from the use of the symmetry of A and orthogonality of the v_i in Arnoldi's algorithm

$$\begin{aligned} \gamma_j &\equiv h_{j-1,j} \\ &= v_{j-1}^T A v_j/\rho_{j-1} = v_j^T A v_{j-1}/\rho_{j-1} \\ &= v_j^T(\beta_j v_j + \alpha_{j-1} v_{j-1} + \gamma_{j-1} v_{j-2})/\rho_{j-1} \\ &= \beta_j \rho_j/\rho_{j-1}. \end{aligned} \tag{13}$$

Note by using the matrix representation (6), the symmetry of A gave (12) immediately, while the usual vector approach required more effort. It was shown in [Pai72] that (10) is an efficient and more numerically reliable way of computing γ_j than (13).

In matrix form the result is, using tridiagonal $T_k \equiv H_k$,

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T = V_{k+1} T_{k+1,k} \tag{14}$$

$$V_k^T V_k = D_k, \quad V_k^T v_{k+1} = 0, \quad V_k^T A V_k = D_k T_k \equiv S_k, \quad \text{say}, \tag{15}$$

$$T_k \equiv \begin{bmatrix} \alpha_1 & \gamma_2 & & \\ \beta_2 & \alpha_2 & \cdot & \\ & \cdot & \cdot & \gamma_k \\ & & \beta_k & \alpha_k \end{bmatrix}, \quad T_{k+1,k} \equiv \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix}, \tag{16}$$

$$S_k \equiv \begin{bmatrix} \rho_1\alpha_1 & \rho_1\gamma_2 & & \\ \rho_2\beta_2 & \rho_2\alpha_2 & \cdot & \\ & \cdot & \cdot & \rho_{k-1}\gamma_k \\ & & \rho_k\beta_k & \rho_k\alpha_k \end{bmatrix}, \tag{17}$$

where obviously $S_k \equiv V_k^T A V_k$ is symmetric by (12). Again, if we choose β_{j+1} to enforce orthonormality of the vectors, we notice $\rho_j = 1$ for $j = 1, 2, \ldots$ gives $\gamma_j = \beta_j$. The *normalized symmetric Lanczos process* is now

$$\begin{aligned} \beta_0 v_0 &\equiv 0 \\ \beta_1 v_1 &= v, \quad v_1^T v_1 = 1, \\ \text{for} \quad j &= 1, \ldots, k \\ & \alpha_j = v_j^T A v_j \\ & \beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}, \\ & \text{stop if} \quad \beta_{j+1} v_{j+1} = 0 \quad \text{else make} \quad v_{j+1}^T v_{j+1} = 1. \end{aligned} \tag{18}$$

And in matrix form this is

$$A V_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T = V_{k+1} T_{k+1,k} \tag{19}$$

$$V_k^T V_k = I, \quad V_k^T v_{k+1} = 0, \quad V_k^T A V_k = T_k, \tag{20}$$

$$T_k \equiv \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \cdot & \\ & \cdot & \cdot & \beta_k \\ & & \beta_k & \alpha_k \end{bmatrix}, \quad T_{k+1,k} \equiv \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix}.$$

The Lanczos process (which is closely related to the 3-term recurrence for orthogonal polynomials) was suggested by Lanczos [Lan50] for finding eigenvalues of symmetric A, and also used in Lanczos' [Lan52] variant of the Conjugate Gradients method (see also [HS52]) for solving $Ax = b$ with symmetric positive definite A.

Since the difference between the above two versions is only one of normalization, they will have the same sort of numerical behaviour — which is surprisingly effective. Orthogonality of the v_j is soon lost, but eigenvalues are still found to almost machine precision by the Lanczos method (see for example [Pai71], and later works by him and many others), and solution of equations with symmetric positive definite A is also very effective using either the [HS52] CG algorithm (see Reid [Rei71] and many later works) or the [Lan52] variant, see [PS75].

2.4 The Lanczos process for unsymmetric A

Clearly (18) is far more efficient than (1) with (7), and we would like a similar simplification with unsymmetric A. Lanczos [Lan50] suggested such a process. This produces two mutually orthogonal sequences of vectors, $\{v_j\}_1^k$ and $\{\tilde{v}_j\}_1^k$ spanning the subspaces $\mathcal{K}^k(A;v)$ and $\mathcal{K}^k(A^T;\tilde{v})$ respectively. These vectors will be seen to produce a tridiagonal matrix from unsymmetric A. This is achieved by considering the process (1) applied to A^T and $\tilde{v}$, as well as to A and v.

The resulting equivalents of the Krylov process (3) are, with obvious notation, after k steps

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T, \tag{21}$$

$$A^T \tilde{V}_k = \tilde{V}_k \tilde{H}_k + \tilde{h}_{k+1,k} \tilde{v}_{k+1} e_k^T, \tag{22}$$

but now we seek elements $h_{i,j}$ and $\tilde{h}_{i,j}$ which make $\tilde{V}_{k+1}^T V_{k+1}$ diagonal. Again we assume we stop whenever we reach an invariant subspace. However we will see this process will actually break down at the first case of $\tilde{v}_i^T v_i = 0$, so we will assume v and $\tilde{v}$ are (fortuitously) chosen to avoid this.

An inductive argument will show how this mutual orthogonality may be

obtained. Suppose after step $k-1$ we have diagonal and nonsingular

$$\tilde{V}_k^T V_k = D_k = \mathrm{diag}(\rho_1, \ldots, \rho_k), \tag{23}$$

(almost all choices of v and $\tilde{v}$ will ensure this for $k = 1$), then with (21) and (22)

$$\begin{aligned} \tilde{V}_k^T A V_k &= D_k H_k + h_{k+1,k} \tilde{V}_k^T v_{k+1} e_k^T & (24) \\ &= \widetilde{H}_k^T D_k + e_k \tilde{h}_{k+1,k} \tilde{v}_{k+1}^T V_k. & (25) \end{aligned}$$

Since H_k and $\widetilde{H}_k$ are upper Hessenberg, equating the right hand sides shows that for $j = 1, \ldots, k-2$, taking $h_{j,k} = 0$ gives $\tilde{v}_j^T v_{k+1} = 0$, and taking $\tilde{h}_{j,k} = 0$ gives $v_j^T \tilde{v}_{k+1} = 0$. Next the (k,k) elements of (24) and (25) show $\tilde{v}_k^T v_{k+1} = \tilde{v}_{k+1}^T v_k = 0$ if we take

$$h_{k,k} = \tilde{h}_{k,k} = \tilde{v}_k^T A v_k / \rho_k \equiv \alpha_k, \quad \text{say.} \tag{26}$$

Next the $(k, k-1)$ element of (25) shows $\tilde{v}_{k+1}^T v_{k-1} = 0$ if we take

$$\tilde{h}_{k-1,k} = \tilde{v}_k^T A v_{k-1} / \rho_{k-1} \equiv \tilde{\gamma}_k, \quad \text{say,} \tag{27}$$

while the $(k, k-1)$ element of (24) shows for the normalizer of v_k that

$$h_{k,k-1} = \tilde{v}_k^T A v_{k-1} / \rho_k \equiv \beta_k, \quad \text{say.} \tag{28}$$

Finally the $(k-1, k)$ element of (24) shows $\tilde{v}_{k-1}^T v_{k+1} = 0$ if we take

$$h_{k-1,k} = \tilde{v}_{k-1}^T A v_k / \rho_{k-1} \equiv \gamma_k, \quad \text{say,} \tag{29}$$

while the $(k-1, k)$ element of (25) shows for the normalizer of $\tilde{v}_k$ that

$$\tilde{h}_{k,k-1} = \tilde{v}_{k-1}^T A v_k / \rho_k \equiv \tilde{\beta}_k, \quad \text{say.} \tag{30}$$

These show that $T_k \equiv H_k$ and $\widetilde{T}_k \equiv \widetilde{H}_k$ are tridiagonal, and also show why nonzero $\tilde{v}_i^T v_i$ is required to avoid breakdowns. If now $\tilde{v}_{k+1}^T v_{k+1}$ is nonzero we

have $\tilde{V}_{k+1}^T V_{k+1}$ diagonal and nonsingular, and we can continue the process. Note the process has not been fully defined, as (at each step) the normalizers $\beta_{k+1} \equiv h_{k+1,k}$ of v_{k+1}, and $\tilde{\beta}_{k+1} \equiv \tilde{h}_{k+1,k}$ of $\tilde{v}_{k+1}$ may be chosen arbitrarily nonzero. Here we consider two possibilities.

By the *standard unsymmetric Lanczos process* we will mean the Lanczos process where β_j is arbitrarily chosen, for example to give $v_j^T v_j = 1, \quad j = 1, 2, \ldots$, but whatever the choice of β_j, $\tilde{\beta}_j$ must be chosen to give

$$\tilde{v}_j^T v_j = 1, \quad j = 1, 2, \ldots$$

so that $D_k = I$ and from (24) and (25) $T_k \equiv H_k = \widetilde{H}_k^T$, giving (see also (27) and (28), and (29) and (30))

$$\tilde{\gamma}_j = \beta_j, \qquad \tilde{\beta}_j = \gamma_j, \qquad j = 1, 2, \ldots,$$

and for (21) and (22)

$$V_k \equiv \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix}, \quad \tilde{V}_k \equiv \begin{bmatrix} \tilde{v}_1 & \tilde{v}_2 & \cdots & \tilde{v}_k \end{bmatrix},$$

$$T_k \equiv \begin{bmatrix} \alpha_1 & \gamma_2 & & \\ \beta_2 & \alpha_2 & \cdot & \\ & \cdot & \cdot & \gamma_k \\ & & \beta_k & \alpha_k \end{bmatrix},$$

$$T_{k+1,k} \equiv \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix}, \quad T_{k,k+1} \equiv \begin{bmatrix} T_k \,\Big|\, \gamma_{k+1} e_k \end{bmatrix},$$

$$\begin{aligned} AV_k &= V_k T_k + \beta_{k+1} v_{k+1} e_k^T = V_{k+1} T_{k+1,k}, \\ A^T \tilde{V}_k &= \tilde{V}_k T_k^T + \gamma_{k+1} \tilde{v}_{k+1} e_k^T = \tilde{V}_{k+1} T_{k,k+1}^T, \\ \tilde{V}_k^T V_k &= I, \quad \tilde{V}_k^T v_{k+1} = V_k^T \tilde{v}_{k+1} = 0, \\ \tilde{V}_k^T A V_k &= T_k, \quad V_k^T A^T \tilde{V}_k = T_k^T. \end{aligned} \tag{31}$$

The detailed algorithm is then (where $\beta_1, \ldots, \beta_{j+1}$ may be chosen arbitrarily)

$$
\begin{aligned}
v_0 &\equiv 0, \quad \tilde{v}_0 \equiv 0 \\
\beta_1 v_1 &= v, \quad \gamma_1 \tilde{v}_1 = \tilde{v} \quad \text{so that} \quad \tilde{v}_1^T v_1 = 1, \\
\text{for} \quad j &= 1, \ldots, k \\
&\qquad \alpha_j = \tilde{v}_j^T A v_j \\
&\qquad \beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \gamma_j v_{j-1} \\
&\qquad \gamma_{j+1} \tilde{v}_{j+1} = A^T \tilde{v}_j - \alpha_j \tilde{v}_j - \beta_j \tilde{v}_{j-1} \\
&\qquad \text{stop if} \quad \beta_{j+1} v_{j+1} = 0 \quad \text{or} \quad \gamma_{j+1} \tilde{v}_{j+1} = 0 \quad \text{or} \\
&\qquad\quad \beta_{j+1} v_{j+1}^T \tilde{v}_{j+1} \gamma_{j+1} = 0, \\
&\qquad \text{otherwise choose} \quad \gamma_{j+1} \quad \text{to give} \quad \tilde{v}_{j+1}^T v_{j+1} = 1. \qquad (32)
\end{aligned}
$$

By the *alternative unsymmetric Lanczos process* we will mean the (apparently new variant of the) Lanczos process where β_j is arbitrarily chosen, but then we take

$$\tilde{\beta}_j = \beta_j, \quad j = 1, 2, \ldots \qquad (33)$$

With this choice it follows from (28) and (30) that

$$\tilde{v}_{k-1}^T A v_k = \tilde{v}_k^T A v_{k-1}, \qquad (34)$$

so from (27) and (29)

$$\tilde{\gamma}_k = \tilde{v}_{k-1}^T A v_k / \rho_{k-1} = \gamma_k, \qquad (35)$$

giving

$$T_k \equiv H_k = \overline{H}_k = \begin{bmatrix} \alpha_1 & \gamma_2 & & \\ \beta_2 & \alpha_2 & \cdot & \\ & \cdot & \cdot & \gamma_k \\ & & \beta_k & \alpha_k \end{bmatrix} \qquad (36)$$

$$
\begin{aligned}
T_{k+1,k} &= \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix} \\
AV_k &= V_k T_k + \beta_{k+1} v_{k+1} e_k^T = V_{k+1} T_{k+1,k}, && (37) \\
A^T \tilde{V}_k &= \tilde{V}_k T_k + \beta_{k+1} \tilde{v}_{k+1} e_k^T = \tilde{V}_{k+1} T_{k+1,k}, && (38) \\
\tilde{V}_k^T V_k &= D_k = \mathrm{diag}(\rho_1, \ldots, \rho_k). \\
\tilde{V}_k^T A V_k &= D_k T_k \equiv S_k \quad \text{is symmetric.} && (39)
\end{aligned}
$$

The surprising outcome is that the two Krylov processes (with A and A^T) use the coefficients in the same way. At the j-th step we could take, see (26), and (34) and (35),

$$
\begin{aligned}
\rho_j &= \tilde{v}_j^T v_j \\
\alpha_j &= \tilde{v}_j^T A v_j / \rho_j && (40) \\
\gamma_j &= \tilde{v}_{j-1}^T A v_j / \rho_{j-1} && (41) \\
\beta_{j+1} v_{j+1} &= A v_j - \alpha_j v_j - \gamma_j v_{j-1} && (42) \\
\beta_{j+1} \tilde{v}_{j+1} &= A^T \tilde{v}_j - \alpha_j \tilde{v}_j - \gamma_j \tilde{v}_{j-1} && (43) \\
\beta_{j+1} &\neq 0 \quad \text{arbitrary.}
\end{aligned}
$$

Again, as with (5) in our first approach to the Arnoldi algorithm, and our comments on the Lanczos process for symmetric A, we find that the above computation is unnecessarily numerically inaccurate. Thus we use the following alternative to (41).

$$
\gamma_j = \beta_j \rho_j / \rho_{j-1} \quad \text{from (29), (30) and (33),} \qquad (44)
$$

which is identical to the choice (10) recommended for the symmetric case.

This alternative unsymmetric Lanczos process is actually just a diagonally scaled version of the standard unsymmetric Lanczos process. This is clear since both processes produce mutually orthogonal vectors spanning the same subspaces — the only difference is the first process produces

$\tilde{V}_k^T V_k = I$, the second one does not. Although the standard process is usually considered the underlying process in the BiCG method, we will use the alternative process to produce a pleasingly simple and clear derivation of BiCG in § 4.6.

Apart from this surprising new variant of the unsymmetric Lanczos process, so far the only original work here has been the matrix based derivation of these algorithms. This is much briefer and, we think, easier to follow than the usual approach, see for example [GL89, § 10.2], but the originality is in the approach alone. However we can now obtain a minor insight into the computational behaviour of these two variants of the unsymmetric Lanczos process.

If A is symmetric, the unsymmetric Lanczos process becomes the symmetric Lanczos process (taking $\tilde{v} = v$ of course, which appears to be the only efficient choice - halving the work). In this case the main step of the unsymmetric Lanczos process with orthonormal vectors (32) becomes

$$\begin{aligned} \alpha_j &= v_j^T A v_j \\ \beta_{j+1} v_{j+1} &= A v_j - \alpha_j v_j - \beta_j v_{j-1}, \\ \text{since} \quad \gamma_j &= \beta_j, \end{aligned}$$

which is just the normalized symmetric Lanczos process (18).

Similarly the alternative unsymmetric Lanczos process becomes

$$\begin{aligned} \rho_j &= v_j^T v_j \\ \alpha_j &= v_j^T A v_j / \rho_j \\ \gamma_j &= \beta_j \rho_j / \rho_{j-1} \\ \beta_{j+1} v_{j+1} &= A v_j - \alpha_j - \gamma_j v_{j-1} \\ \beta_{j+1} &\neq 0 \quad \text{arbitrary}, \end{aligned}$$

which is the same as the arbitrarily normalized Lanczos process for sym-

metric A. Thus we can hope that these two unsymmetric processes will maintain some of the well-known good numerical properties of their equivalent symmetric processes.

3 Preliminaries – The QR decomposition

Some of the algorithms presented in this paper, for example LSQR, GMRES(m) and QMR, require the reduction of a sequence of matrices, each of which contains the previous as leading submatrix, to upper triangular form. To this end, one may perform a sequence of QR decompositions. The notation and method presented in this section will be used in those algorithms for this purpose. For the decomposition one may choose from 2×2 rotation matrices, 2×2 reflection matrices (2×2 Householder transformations), or fast Givens transformations. For simplicity, 2×2 rotation matrices were used here. The matrices arising in these algorithms are of a specific shape – in general upper Hessenberg, more specifically tridiagonal or lower bidiagonal, with an additional row containing a nonzero element in the last column only. The upper triangular result may in these special cases be upper tridiagonal or upper bidiagonal. For the general matrix

$$H_{k+1,k} \equiv \begin{bmatrix} h_{11} & h_{12} & \cdot & h_{1k} \\ h_{21} & h_{22} & \cdot & h_{2k} \\ & \cdot & \cdot & \cdot \\ & & h_{k,k-1} & h_{kk} \\ & & & h_{k+1,k} \end{bmatrix} \in R^{(k+1)\times k}$$

(h_{ij} may be zero for $j > i+1$ or $j > i$) the decompostion is

$$Q_k^T H_{k+1,k} = \begin{bmatrix} R_k \\ 0 \end{bmatrix},$$

where $Q_k \in R^{(k+1)\times(k+1)}$ is orthogonal,

and $R_k \in R^{k\times k}$ is upper triangular.

This is achieved using rotations embedded at the i,i position of $k+1$-dimensional identity matrices, as

$$Q_{i,k} \equiv \left[\begin{array}{c|cc|c} I_{i-1} & & & \\ \hline & c_i & s_i & \\ & -s_i & c_i & \\ \hline & & & I_{k-i} \end{array}\right] \in \mathcal{R}^{(k+1)\times(k+1)}$$

$$\text{such that } Q_k^T \equiv Q_{k,k}^T \cdots Q_{1,k}^T$$

$$\equiv Q_{k,k}^T \left[\begin{array}{cc} Q_{k-1}^T & \\ & 1 \end{array}\right].$$

The effect of the appropriately embedded Q_{k-1}^T on $H_{k+1,k}$ is

$$\left[\begin{array}{cc} Q_{k-1}^T & \\ & 1 \end{array}\right] \left[\begin{array}{c|c} H_{k,k-1} & \begin{array}{c} h_{1k} \\ \vdots \\ h_{kk} \end{array} \\ \hline 0 & h_{k+1,k} \end{array}\right] = \left[\begin{array}{c|c} R_{k-1} & \begin{array}{c} r_{1k} \\ \vdots \\ r_{k-1,k} \end{array} \\ \hline 0 & \tilde{r}_{kk} \\ \hline 0 & h_{k+1,k} \end{array}\right],$$

while $Q_{k,k}^T$, which alters only rows k and $k+1$, zeros $h_{k+1,k}$

$$\left[\begin{array}{cc} c_k & -s_k \\ s_k & c_k \end{array}\right] \left[\begin{array}{c} \tilde{r}_{kk} \\ h_{k+1,k} \end{array}\right] = \left[\begin{array}{c} r_{kk} \\ 0 \end{array}\right],$$

$$\Rightarrow r_{kk} = \sqrt{\tilde{r}_{kk}^2 + h_{k+1,k}^2}$$

$$c_k = \tilde{r}_{kk}/r_{kk}$$

$$s_k = -h_{k+1,k}/r_{kk}.$$

The orthogonality of the $Q_{i,k}$ leads to that of Q_k.

A scalar multiple of the first column of Q_k^T is given by

$$g_k \equiv \beta Q_k^T e_1 \equiv \beta Q_{k,k}^T \cdots Q_{1,k}^T e_1 \equiv \left[\begin{array}{c} f_k \\ \bar{\phi}_{k+1} \end{array}\right] \equiv \left[\begin{array}{c} \phi_1 \\ \vdots \\ \phi_k \\ \bar{\phi}_{k+1} \end{array}\right], \qquad (45)$$

where

$$
\begin{aligned}
f_k &= \begin{bmatrix} f_{k-1} \\ \phi_k \end{bmatrix}, \\
f_1 = \phi_1 &= \beta c_1 \\
\bar{\phi}_2 &= \beta s_1 \\
\phi_k &= c_k \bar{\phi}_k = \beta c_k s_{k-1} \cdots s_1 \\
\bar{\phi}_{k+1} &= s_k \bar{\phi}_k = \beta s_k \cdots s_1. \qquad (46)
\end{aligned}
$$

Note: This notation – $Q_k, Q_{i,k}, g_k, f_k, \bar{\phi}_k, \phi_k, c_k, s_k$ – is used throughout this document with exactly these meanings, except in algorithms SYMMLQ and MINRES where the Q_k, c_k, s_k refer to the LQ decomposition of the matrices involved.

Reference: for example [GL89, § 5.1 5.2]

4 Algorithms for Solving Linear Systems

We now consider most of the more common Krylov subspace methods for solving linear systems of equations. Each of the following algorithms has been converted into the matrix formalism presented in § 1.1. Each subsection is begun with the problem statement, either large systems of linear equations, or the linear least squares problem, that is

$$Ax = b, \quad A \in R^{n\times n} \quad \text{or} \quad \min_{x\in R^n} \|b - Ax\|, \quad A \in R^{m\times n}.$$

The type of matrix A, for example symmetric or not, positive definite or indefinite, is given.

We examine different methods, where sometimes these are just different variants of one method. We continue to use the word method rather than implementation or algorithm, as there could also be more than one implementation for a given variant. For each such method the process used to form vectors $\{v_1, \ldots, v_k\}$ spanning some associated subspace is named, as is the subspace.

Next for each method the subproblem arises as a result of considering successive solutions of the form

$$x_k = x_0 + V_k y_k$$

for some initial guess x_0. For simplicity, x_0 will usually[1] be taken as zero. So the iterates look like

$$\begin{aligned} x_k &= V_k y_k \in \mathcal{R}(V_k), \\ \text{where } V_k &\equiv \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix}, \end{aligned} \tag{47}$$

and the way in which this subproblem arises and is solved is described.

[1] Except in GMRES(m) where with each restart a new nonzero x_0 is used.

For some methods, where it is important for the development here, we will describe the basic implementation of the method, showing how the vectors from the process can be combined via the solution of the subproblem to produce an efficient (although not yet optimized) algorithm.

Then, to facilitate the most likely implementation of a stopping condition for each algorithm, the norm of the *residual* of the current iterate

$$r_k \equiv b - Ax_k = b - AV_k y_k \tag{48}$$

is considered.

Finally (when we can, and when it is appropriate) we relate this matrix derivation to one or more of the standard vector implementations used today, thus closing the circle by showing how what are usually confusing (to the non-expert) vector presentations can be easily derived and well-motivated using the matrix approach recommended here.

4.1 The Lanczos (Conjugate Gradients) Method

Problem: Solve $Ax = b$ for x when A is symmetric positive definite (SPD).

First we derive what will later be shown to be the Conjugate Gradients approximations x_k from the normalized symmetric Lanczos process (18). We refer to this as:

The unit norm Lanczos-CG method.

Process: the normalized Lanczos process (18) for symmetric A to produce orthonormal $\{v_1, \ldots, v_k\}$ spanning $\mathcal{K}^k(A; b)$, starting with $v_1 = b/\beta_1$, where $\beta_1 \equiv \|b\|$.

To derive the subproblem we use (19) and consider iterates of the form $x_k = V_k y_k$ as suggested in (47), and take $T_k y_k = \beta_1 e_1$, so that

$$\begin{aligned} r_k &= b - Ax_k = b - AV_k y_k \\ &= b - V_k T_k y_k - \beta_{k+1} v_{k+1} e_k^T y_k \\ &= V_k(\beta_1 e_1 - T_k y_k) - \beta_{k+1} v_{k+1} e_k^T y_k \\ &= -\beta_{k+1} v_{k+1} e_k^T y_k. \end{aligned} \tag{49}$$

In theory $\beta_{k+1} v_{k+1} = 0$ for some $k \leq n$, the dimension of A, and the problem will be solved.

Subproblem: Solve $T_k y_k = \beta_1 e_1$, where T_k is symmetric tridiagonal.

This has shown in simple matrix terms how the Lanczos process (19) $AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T$ and the subproblem $T_k y_k = \beta_1 e_1$ give approximations $x_k = V_k y_k$ to the solution of $Ax = b$. However we now have to combine these two (Process and Subproblem) to produce a useful *implementation*.

Implementation: For A SPD, $T_k = V_k^T A V_k$ is as well, therefore the subproblem always has a solution and it is possible to perform a Cholesky factorisation of $T_k = L_k L_k^T$, where L_k is lower bidiagonal. In order to obtain objects c_k, w_k that can be computed sequentially and discarded after

use — an important efficiency for large sparse problems — define

$$z_k \equiv \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_k \end{bmatrix} \equiv L_k^T y_k, \quad W_k \equiv \begin{bmatrix} w_1 & w_2 & \cdots & w_k \end{bmatrix} \equiv V_k L_k^{-T},$$

so the system is transformed to

$$\begin{aligned} T_k y_k = L_k L_k^T y_k = L_k z_k \quad &\Rightarrow \quad L_k z_k = \beta_1 e_1, \\ V_k y_k = V_k L_k^{-T} L_k^T y_k \quad &\Rightarrow \quad x_k = W_k z_k. \end{aligned}$$

Now, L_k is formed one row at a time:

$$\begin{aligned} T_k &= \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1} & 0 \\ \hline 0 & \delta_k & \gamma_k \end{array}\right] \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1}^T & \delta_k \\ \hline 0 & 0 & \gamma_k \end{array}\right] \\ &= \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1}L_{k-1}^T & \gamma_{k-1}\delta_k \\ \hline 0 & \delta_k\gamma_{k-1} & \delta_k^2+\gamma_k^2 \end{array}\right] = \left[\begin{array}{cc|c} & & 0 \\ & T_{k-1} & \beta_k \\ \hline 0 & \beta_k & \alpha_k \end{array}\right] \\ &\Rightarrow \quad \gamma_1 = \sqrt{\alpha_1}, \\ &\qquad \delta_k = \beta_k/\gamma_{k-1}, \quad \gamma_k = \sqrt{\alpha_k - \delta_k^2}, \quad k > 1. \end{aligned} \tag{50}$$

And so z_k is formed one entry at a time:

$$\begin{aligned} \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1} & 0 \\ \hline 0 & \delta_k & \gamma_k \end{array}\right] \begin{bmatrix} z_{k-1} \\ \zeta_k \end{bmatrix} &= \begin{bmatrix} L_{k-1}z_{k-1} \\ \delta_k\zeta_{k-1} + \gamma_k\zeta_k \end{bmatrix} = \beta_1 e_1 \\ \Rightarrow \zeta_1 = \beta_1/\gamma_1, \quad \zeta_k &= -\delta_k\zeta_{k-1}/\gamma_k, \quad k > 1. \end{aligned} \tag{51}$$

W_k is also formed one column at a time ($W_k L_k^T = V_k$):

$$\begin{bmatrix} W_{k-1} & w_k \end{bmatrix} \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1}^T & \delta_k \\ \hline 0 & 0 & \gamma_k \end{array}\right] = \begin{bmatrix} W_{k-1}L_{k-1}^T & \delta_k w_{k-1} + \gamma_k w_k \end{bmatrix}$$

$$\Rightarrow w_1 = v_1/\gamma_1, \quad w_k = (v_k - \delta_k w_{k-1})/\gamma_k, \quad k > 1.$$

Finally,

$$\begin{aligned} x_k &= W_k z_k = \begin{bmatrix} W_{k-1} & w_k \end{bmatrix} \begin{bmatrix} z_{k-1} \\ \zeta_k \end{bmatrix} \\ &= W_{k-1} z_{k-1} + \zeta_k w_k = x_{k-1} + \zeta_k w_k, \quad k \geq 1. \end{aligned}$$

Again we see how each step of the implementation makes obvious sense when approached via this matrix formulation.

For large sparse problems we would like to stop in $k << n$ steps, and since a solution with sufficiently small residual is what is often wanted, we look at how the residual norm behaves here.

Residual Norm: The following estimation of the norm of the residual r_k, as defined in (49) can be used.

$$\begin{aligned} r_k &= -\beta_{k+1} v_{k+1} e_k^T y_k = -\beta_{k+1} v_{k+1} y_{kk} \\ \|r_k\| &= \beta_{k+1} |y_{kk}|, \quad \text{where } y_{kk} \text{ is the } k\text{-th entry of } y_k. \end{aligned}$$

Since y_k is not computed directly, one must consider

$$\begin{aligned} L_k^T y_k &= z_k \\ \left[\begin{array}{c|c} L_{k-1}^T & \begin{matrix} 0 \\ \delta_k \end{matrix} \\ \hline \begin{matrix} 0 & 0 \end{matrix} & \gamma_k \end{array}\right] \begin{bmatrix} y_{k1} \\ \vdots \\ y_{kk} \end{bmatrix} &= \begin{bmatrix} \vdots \\ \gamma_k y_{kk} \end{bmatrix} = \begin{bmatrix} z_{k-1} \\ \zeta_k \end{bmatrix} \\ y_{kk} &= \zeta_k/\gamma_k \\ \Rightarrow \|r_k\| &= \beta_{k+1} |\zeta_k/\gamma_k| \\ &= \beta_{k+1} |\delta_k \zeta_{k-1}/\gamma_k^2| \quad \text{by (51)} \\ &= \beta_{k+1} |\beta_k \zeta_{k-1}/\gamma_{k-1}|/\gamma_k^2 \quad \text{by (50)} \\ &= \beta_{k+1} \|r_{k-1}\|/\gamma_k^2, \quad k \geq 1. \end{aligned}$$

This matrix approach also tends to make analysis of such algorithms fairly easy. For example, since r_j is a scalar multiple of v_{j+1} and $V_k^T V_k = I$, the residuals r_j are orthogonal. Next

$$W_k^T A W_k = L_k^{-1} V_k^T A V_k L_k^{-T} = L_k^{-1} T_k L_k^{-T} = I,$$

so the w_j (the search vectors in moving from x_{j-1} to x_j) are A-orthogonal.

That x_k is the same as the ordinary CG solution follows for example from [PPdV95, § 3], since $x_k = V_k y_k \in \mathcal{K}^k(A;b)$ and $r_k \equiv b - Ax_k \perp \mathcal{K}^k(A;b)$, and so x_k is the Galerkin solution from $\mathcal{K}^k(A;b)$ to $Ax = b$, which is exactly what CG delivers.

This unit norm Lanczos-CG method, based upon the Lanczos process, has the nice property that the vectors $v_1, v_2, \ldots$ are orthonormal. However, let us derive a method which more closely mimics the standard CG method in that $v_1, v_2, \ldots$ are the residual vectors themselves. Because it gives a more direct analogy with the standard CG method, we call it:

The Lanczos-CG method

Process: the Lanczos process (11) for symmetric A, starting with $v_1 = b$. The normalization β_{j+1} is unspecified as yet.

Using (14) and considering iterates of the form $x_k = V_k y_k$ as suggested in (47) and solving for $T_k y_k = e_1$,

$$\begin{aligned} r_k &= b - Ax_k = b - AV_k y_k \\ &= V_k(e_1 - T_k y_k) - \beta_{k+1} v_{k+1} e_k^T y_k \\ &= -\beta_{k+1} y_{kk} v_{k+1}. \end{aligned} \tag{52}$$

Subproblem: Solve $T_k y_k = e_1$, where T_k is tridiagonal, but not necessarily symmetric.

Implementation: We choose to perform an LDU decomposition of $T_k = D_k^{-1} S_k$, see (15). $S_k = V_k^T A V_k$ is SPD, and D_k is a diagonal matrix with

positive elements, so the LDU decomposition without pivoting appears to be a safe choice. There is considerable freedom in such a decomposition, and the one which suits our purpose is as follows. With the as yet unknown $\mathcal{D}_k \equiv -\mathrm{diag}(\beta_2, \ldots, \beta_{k+1})$, write

$$
\begin{aligned}
T_k &= L_k \mathcal{D}_k U_k \\
T_k &= \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1} & 0 \\ \hline 0 & -1 & 1 \end{array}\right] \left[\begin{array}{cc|c} & & 0 \\ & \mathcal{D}_{k-1} & 0 \\ \hline 0 & 0 & -\beta_{k+1} \end{array}\right] \left[\begin{array}{cc|c} & & 0 \\ & U_{k-1} & -\frac{\gamma_k}{\beta_k} \\ \hline 0 & 0 & 1 \end{array}\right] \\
&= \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1}\mathcal{D}_{k-1}U_{k-1} & \gamma_k \\ \hline 0 & \beta_k & -(\gamma_k + \beta_{k+1}) \end{array}\right].
\end{aligned}
$$

Note that a simpler method could be obtained by combining $\mathcal{D}$ and U, and using for example $\mathcal{U} \equiv \mathcal{D}U$ instead, but the approach here will produce the equivalents of various elements of CG that we want.

We can now determine our normalizers β_{j+1} by equating the right side of the above equation with T_k in (16). Clearly

$$
\begin{aligned}
\alpha_1 &= -\beta_2; \qquad \alpha_k = -(\gamma_k + \beta_{k+1}), \quad k > 1, \\
\Rightarrow \beta_2 &\equiv -\alpha_1; \qquad \beta_{k+1} \equiv -(\alpha_k + \gamma_k), \quad k > 1.
\end{aligned} \tag{53}
$$

We will show below that this $\beta_{j+1} \neq 0$, and so is a satisfactory normalizer for the process, and U_{k+1} exists, but first we complete this implementation. Define

$$
z_k \equiv \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_k \end{bmatrix} \equiv U_k y_k, \quad W_k \equiv \begin{bmatrix} w_1 & w_2 & \cdots & w_k \end{bmatrix} \equiv V_k U_k^{-1}, \tag{54}
$$

so $L_k \mathcal{D}_k z_k = e_1$ is our subproblem, and $x_k = W_k z_k$ is our solution. Solving

$$\begin{aligned} L_k \mathcal{D}_k z_k &= \begin{bmatrix} -\beta_2 \zeta_1 \\ \beta_2 \zeta_1 - \beta_3 \zeta_2 \\ \vdots \\ \beta_k \zeta_{k-1} - \beta_{k+1} \zeta_k \end{bmatrix} = e_1 \\ \Rightarrow \zeta_1 &= -\frac{1}{\beta_2} \\ \zeta_k &= \frac{\beta_k}{\beta_{k+1}} \zeta_{k-1} = \frac{\beta_{k-1}}{\beta_{k+1}} \zeta_{k-2} = \cdots = -\frac{1}{\beta_{k+1}}, \quad k > 1. \end{aligned} \tag{55}$$

Solving $W_k U_k = V_k$ gives

$$w_1 = v_1, \quad w_k = v_k + \frac{\gamma_k}{\beta_k} w_{k-1}, \quad k > 1.$$

Finally

$$x_k = W_{k-1} z_{k-1} + \zeta_k w_k = x_{k-1} + \zeta_k w_k, \quad k \geq 1.$$

Residual Norm:

$$\begin{aligned} r_k &= -\beta_{k+1} y_{kk} v_{k+1} \qquad \text{by (52)} \\ &= -\beta_{k+1} \zeta_k v_{k+1} \quad \text{by (54)} \\ &= -\beta_{k+1} \times -\frac{1}{\beta_{k+1}} v_{k+1} \quad \text{by (55)} \\ &= v_{k+1}{}^{L} \qquad (56) \\ \Rightarrow \|r_k\| &= \|v_{k+1}\|. \end{aligned}$$

So the Lanczos process (11) for symmetric A, with $v_1 = b$, can be used to produce the (CG) residual vectors $r_k = b - Ax_k$ themselves, as long as β_{j+1} is chosen as in (53).

It remains for us to prove that no β_{j+1} found this way can be zero. Since A is SPD, α_1 is nonzero and so β_2 is nonzero. Suppose $\beta_2, \ldots, \beta_j$ obtained from (53) are nonzero. Then with $e^T \equiv [1, \ldots, 1]$, $e^T T_j = [0, \ldots, 0, \alpha_j + \gamma_j]$,

which must be nonzero since T_j is nonsingular. Thus no β_{j+1} will be zero, and this algorithm will not break down if A is SPD. It is also clear that the $L_k\mathcal{U}_k$ 'factorization' is numerically well behaved, with the only computation being $\beta_{j+1} = -(\alpha_j + \gamma_j)$, and there is no large element growth in $\mathcal{U}_k$.

Equivalence of the Lanczos-CG method with CG:

This Lanczos-CG method is just a scaling of the unit norm Lanczos-CG method, and provides the same solution in a marginally different way. It is therefore also the CG solution. However we would like to compare the resulting algorithms closely, so we give a proof which shows clearly what equivalences occur.

The CG Method — slightly altered from 'Templates' [BBC+94] — is as follows.

$r_0 = b$

for $i = 1, 2, \ldots$

$\rho_{i-1} = r_{i-1}^T r_{i-1}$

if $(i = 1)$

$p_i = r_{i-1}$

else

$\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$

$p_i = r_{i-1} + \beta_{i-1} p_{i-1}$

$q_i = Ap_i$

$\alpha_i = \rho_{i-1}/p_i^T q_i$

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

$r_i = r_{i-1} - \alpha_i q_i$

check convergence; continue if necessary

end

To avoid confusion we will use superscript ^{C} to distinguish 'Templates' terms from those here, where it will also make life easier if we re-index two

of the 'Templates' terms as follows:

$$\begin{aligned} \rho_i^C &\equiv \text{'Templates' } \rho_{i-1}, \\ \beta_i^C &\equiv \text{'Templates' } \beta_{i-1}. \end{aligned}$$

We now write the algorithms side by side for comparison, where in the Lanczos-CG algorithm we replace v_{i+1} by r_i, see (56), and introduce $u_i \equiv Av_i$. To handle the $i = 1$ case we define

CG	Lanczos-CG
$\rho_0^C = 1$	$\rho_0 = 1, \quad \beta_1 = 0$
$x_0^C = 0, \quad p_0^C = 0$	$x_0 = 0, \quad w_0 = 0, \quad r_{-1} = 0$

Then for $i = 1, 2, 3, \ldots$ we have

CG	Lanczos-CG
$\rho_i^C = r_{i-1}^{CT} r_{i-1}^C$	$\rho_i = r_{i-1}^T r_{i-1}$
$\beta_i^C = \rho_i^C / \rho_{i-1}^C$	$\gamma_i = \beta_i \rho_i / \rho_{i-1}$
$p_i^C = r_{i-1}^C + \beta_i^C p_{i-1}^C$	$w_i = r_{i-1} + (\gamma_i / \beta_i) w_{i-1}$
$q_i^C = A p_i^C$	$u_i = A r_{i-1}$
$\alpha_i^C = \rho_i^C / p_i^{CT} q_i^C$	$\alpha_i = r_{i-1}^T u_i / \rho_i$
	$\beta_{i+1} = -(\alpha_i + \gamma_i)$
$x_i^C = x_{i-1}^C + \alpha_i^C p_i^C$	$x_i = x_{i-1} - (1/\beta_{i+1}) w_i$
$r_i^C = r_{i-1}^C - \alpha_i^C q_i^C$	$r_i = (u_i - \alpha_i r_{i-1} - \gamma_i r_{i-2}) / \beta_{i+1}$

The only significant difference in cost occurs in the last line — n extra multiplications, subtractions and divisions in the Lanczos-CG algorithm. However note the Lanczos-CG here was written for comparison, not efficiency. The equivalences are

$$\begin{aligned} \rho_i &= \rho_i^C, \quad \gamma_i / \beta_i = \beta_i^C, \quad w_i = p_i^C \\ -1\beta_{i+1} &= \alpha_i^C, \quad x_i = x_i^C, \quad r_i = r_i^C. \end{aligned} \tag{57}$$

A straightforward examination shows this is true for $i = 1$. Suppose it is true for $i = 1, 2, \ldots, k-1$. Then it is clear that

$$\rho_k = \rho_k^C, \quad \gamma_k/\beta_k = \beta_k^C, \quad w_k = p_k^C.$$

We now need only prove

$$-1/\beta_{k+1} = \alpha_k^C, \tag{58}$$

so that $x_k = x_k^C$, and their residuals r_k and r_k^C must then also be equal, since r_i^C is clearly the residual for x_i^C by examining the last two lines of CG, and we have shown r_i is the residual of x_i in the Lanczos-CG. We can prove (58) using our matrix formulation of the Lanczos-CG algorithm. We know $W_k = V_k U_k^{-1}$, so

$$\begin{aligned} W_k^T A W_k &= U_k^{-T} V_k^T A V_k U_k^{-1} = U_k^{-T} D_k T_k U_k^{-1} \\ &= U_k^{-T} D_k (L_k \mathcal{D}_k U_k) U_k^{-1} = U_k^{-T} D_k L_k \mathcal{D}_k, \end{aligned}$$

which, by the way, is lower triangular and symmetric, and therefore diagonal, showing A-orthogonality again. The (i,i)-element of this is

$$w_i^T A w_i = e_i^T U_k^{-T} (\rho_i e_i - \rho_{i+1} e_{i+1})(-\beta_{i+1}).$$

But $e_i^T U_k^{-T} = (\times, \ldots, \times, 1, 0, \ldots, 0)$ is zero after the i-th element, so since $p_i^C = w_i$, $i = 1, \ldots, k$,

$$p_i^{CT} A p_i^C = w_i^T A w_i = -\rho_i \beta_{i+1}, \qquad i = 1, \ldots, k.$$

From the CG algorithm we then see

$$\alpha_i^C = \rho_i^C / p_i^{CT} q_i^C = \rho_i / p_i^{CT} A p_i^C = -1/\beta_{i+1}, \quad i = 1, \ldots, k,$$

and so (58) holds, and the induction proof of (57) is complete, showing that in theory these algorithms not only compute the same iterates (as was obvious earlier), but also several of the intermediate quantities are

identical. However the algorithms are clearly not identical, as the last line of each clearly highlights, and may have different numerical behaviours. This suggests a matrix development of CG would be useful, but we leave this for future work.

4.2 Symmetric LQ Method (SYMMLQ)

Problem: Solve $Ax = b$ with A nonsingular symmetric, but indefinite.

Process: the normalized Lanczos process (18) for symmetric A on $\mathcal{K}^k(A;b)$ starting with $v_1 = b/\|b\|$.

In this section iterates of the form $x_k^C = V_k y_k^C$ are produced as well as iterates $x_k^L = V_{k+1} y_k^L$. (The superscript C denotes the solution that would be obtained by the method of Conjugate Gradients, while the superscript L refers to the SYMMLQ method.) Using (19) and solving $T_k y_k^C = \beta_1 e_1$, $\beta_1 \equiv \|b\|$, one obtains the result (49) as for the Lanczos method.

Subproblem (Conjugate Gradients method): $T_k y_k^C = \beta_1 e_1$, where the matrix T_k is symmetric tridiagonal. T_k may be singular for some k, in which case this subproblem will not have a (unique) solution, see later, (p. 38, 39).

Implementation: Since A is indefinite, there is a possibility that T_k is as well. Therefore a Cholesky factorisation may break down. Instead, consider an LQ decomposition.

$$T_k \bar{Q}_k^T = \bar{L}_k, \quad \bar{Q}_k^T \bar{Q}_k = I,$$

where one method of factorising has the $\bar{Q}_k^T = Q_{1,k} \cdots Q_{k-1,k}$ as orthogonal products of (reflection) matrices embedded in the k-dimensional identity starting in the (i,i) entry.

$$Q_{i,k} \equiv \left[\begin{array}{c|cc|c} I_{i-1} & & & \\ \hline & c_i & s_i & \\ & s_i & -c_i & \\ \hline & & & I_{k-i-1} \end{array}\right] \in \mathcal{R}^{k \times k} \qquad (59)$$

These are designed to zero the superdiagonal $(i, i+1)$ entry of T_k. $\bar{L}_k$ is the lower tridiagonal result. The notation $\bar{L}_k$ is used to distinguish it from L_k — the leading $k \times k$ part of $\bar{L}_{k+1}$ — from which it differs only in the (k,k)

element ($\bar{\gamma}_k$ becomes γ_k). The first two steps illustrate the process:

$$\begin{bmatrix} T_2 \\ 0 \quad \beta_3 \end{bmatrix} \bar{Q}_2^T = \begin{bmatrix} \alpha_1 & \beta_2 \\ \beta_2 & \alpha_2 \\ 0 & \beta_3 \end{bmatrix} \begin{bmatrix} c_1 & s_1 \\ s_1 & -c_1 \end{bmatrix} = \begin{bmatrix} \gamma_1 & \\ \delta_2 & \bar{\gamma}_2 \\ \epsilon_3 & \bar{\delta}_3 \end{bmatrix} = \begin{bmatrix} \bar{L}_2 \\ \epsilon_3 \quad \bar{\delta}_3 \end{bmatrix}$$

$$T_3 \bar{Q}_3^T = \begin{bmatrix} \gamma_1 & & \\ \delta_2 & \bar{\gamma}_2 & \beta_3 \\ \epsilon_3 & \bar{\delta}_3 & \alpha_3 \end{bmatrix} \begin{bmatrix} 1 & & \\ & c_2 & s_2 \\ & s_2 & -c_2 \end{bmatrix} = \begin{bmatrix} \gamma_1 & & \\ \delta_2 & \gamma_2 & \\ \epsilon_3 & \delta_3 & \bar{\gamma}_3 \end{bmatrix}$$

$$= \left[\begin{array}{cc|c} & L_2 & \begin{matrix} 0 \\ 0 \end{matrix} \\ \hline \epsilon_3 & \delta_3 & \bar{\gamma}_3 \end{array}\right] = \bar{L}_3.$$

Thus after k steps

$$\bar{L}_k = \begin{bmatrix} \gamma_1 & & & & \\ \delta_2 & \gamma_2 & & & \\ \epsilon_3 & \delta_3 & \gamma_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \epsilon_{k-1} & \delta_{k-1} & \gamma_{k-1} \\ & & & \epsilon_k & \delta_k \quad \bar{\gamma}_k \end{bmatrix} = \left[\begin{array}{cc|c} & L_{k-1} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline 0 & \epsilon_k & \delta_k \quad \bar{\gamma}_k \end{array}\right]. \qquad (60)$$

Note that T_k in (20) may be singular, in which case $\bar{L}_k$ is singular. But since $\beta_2 \cdots \beta_k \neq 0$, L_{k-1} is nonsingular. For ease of calculation define

$$\bar{z}_k \equiv \begin{bmatrix} \zeta_1 \\ \vdots \\ \zeta_{k-1} \\ \bar{\zeta}_k \end{bmatrix} \equiv \begin{bmatrix} z_{k-1} \\ \bar{\zeta}_k \end{bmatrix} \equiv \bar{Q}_k y_k^C$$

$$\bar{W}_k \equiv \begin{bmatrix} w_1 & \cdots & w_{k-1} & \bar{w}_k \end{bmatrix} \equiv \begin{bmatrix} W_{k-1} & \bar{w}_k \end{bmatrix} \equiv V_k \bar{Q}_k^T, \qquad (61)$$

so the system

$$T_k y_k^C = \beta_1 e_1, \quad x_k^C = V_k y_k^C$$

is transformed to

$$\bar{L}_k \bar{z}_k = \beta_1 e_1, \quad x_k^C = \bar{W}_k \bar{z}_k.$$

At each step, γ_k, c_k and s_k are calculated:

$$\gamma_k = \sqrt{\bar{\gamma}_k^2 + \beta_{k+1}^2}, \quad c_k = \bar{\gamma}_k/\gamma_k, \quad s_k = \beta_{k+1}/\gamma_k, \tag{62}$$

while the last row of $\bar{L}_{k+1}$ is developed at the same time:

$$\begin{aligned} \bar{\gamma}_{k+1} &= -(\beta_{k+1} c_{k-1} s_k + \alpha_{k+1} c_k), \quad \delta_{k+1} = \alpha_{k+1} s_k - \beta_{k+1} c_{k-1} c_k, \\ \epsilon_{k+1} &= \beta_{k+1} s_{k-1}. \end{aligned} \tag{63}$$

By comparing the last entry of $\bar{L}_k \bar{z}_k$ with the k-th entry of $\bar{L}_{k+1} \bar{z}_{k+1}$ (both should be zero):

$$\bar{\gamma}_k \bar{\zeta}_k = \gamma_k \zeta_k \quad \Rightarrow \quad \zeta_k = \bar{\gamma}_k \bar{\zeta}_k / \gamma_k = c_k \bar{\zeta}_k. \tag{64}$$

Using (59) and (61):

$$\begin{aligned} \bar{w}_1 &= v_1 \\ \begin{bmatrix} \bar{w}_k & v_{k+1} \end{bmatrix} \begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} &= \begin{bmatrix} w_k & \bar{w}_{k+1} \end{bmatrix}, \end{aligned} \tag{65}$$

so passing from $\{\bar{w}_k, v_{k+1}\}$ to $\{w_k, \bar{w}_{k+1}\}$ is painless. The above algorithm is not implemented, since if $\bar{L}_k$ is singular, then $\bar{z}_k$ is undefined. This led Paige and Saunders [PS75] to consider the following alternative subproblem.

Subproblem (SYMMLQ): The minimum 2-norm solution to the problem $T_{k+1,k}^T y_k^L = \beta_1 e_1$, where $T_{k+1,k}$ is as in (19). The matrix $T_{k+1,k}$ has full column rank, and the subproblem always has a unique solution.

This is solved by performing an LQ decomposition of $T_{k+1,k}^T$,

$$\begin{aligned} T_{k+1,k}^T \bar{Q}_{k+1}^T &= \left[\begin{array}{c|c} T_k & \beta_{k+1} e_k \end{array} \right] \left[\begin{array}{c|c} Q_k^T & \bar{q}_{k+1}^T \end{array} \right] = \left[\begin{array}{c|c} L_k & 0 \end{array} \right] \\ \text{solving} \quad L_k z_k &= \beta_1 e_1 \\ \text{and setting} \quad y_k^L &= Q_k^T z_k. \\ \text{Finally} \quad x_k^L &= V_{k+1} y_k^L = V_{k+1} Q_k^T z_k = W_k z_k = x_{k-1}^L + \zeta_k w_k. \end{aligned}$$

Now L_k is nonsingular if $\beta_{k+1} \neq 0$, so z_k does exist and we can safely compute the SYMMLQ solution. Notice x_k^C can easily be computed using

$$x_k^C = \bar{W}_k \bar{z}_k = W_{k-1} z_{k-1} + \bar{\zeta}_k \bar{w}_k = x_{k-1}^L + \bar{\zeta}_k \bar{w}_k. \tag{66}$$

The SYMMLQ solution is more easily updated than x_k^C (when x_k^C exists), but the latter *may* be more desirable at the finish.

Residual Norms: For the Conjugate Gradients solution

$$\begin{aligned}
r_k^C &= b - A\bar{W}_k \bar{z}_k = b - AV_k \bar{Q}_k^T \bar{z}_k \\
&= b - V_{k+1} T_{k+1,k} \bar{Q}_k^T \bar{z}_k \\
&= b - V_{k+1} \left[\begin{array}{c} \bar{L}_k \\ \hline 0 \quad 0 \quad \beta_{k+1} s_{k-1} \quad -\beta_{k+1} c_{k-1} \end{array} \right] \bar{z}_k \qquad (67) \\
&= b - V_{k+1} \{ \beta_1 e_1 + \beta_{k+1} (s_{k-1} \zeta_{k-1} - c_{k-1} \bar{\zeta}_k) e_{k+1} \} \\
&= -\beta_{k+1} (s_{k-1} \zeta_{k-1} - c_{k-1} \bar{\zeta}_k) v_{k+1} \qquad (68) \\
\Rightarrow \|r_k^C\| &= \beta_{k+1} |s_{k-1} \zeta_{k-1} - c_{k-1} \bar{\zeta}_k|.
\end{aligned}$$

Now, for the SYMMLQ residual,

$$\begin{aligned}
r_k^L &= b - Ax_k^L = b - A(x_{k+1}^C - \bar{\zeta}_{k+1} \bar{w}_{k+1}) \quad \text{by (66)} \\
&= r_{k+1}^C + \bar{\zeta}_{k+1} A \bar{w}_{k+1} \\
&= r_{k+1}^C + \bar{\zeta}_{k+1} AV_{k+1} Q_{k+1}^T e_{k+1} \quad \text{by (61)} \\
&= r_{k+1}^C + \bar{\zeta}_{k+1} V_{k+2} T_{k+2,k+1} Q_{k+1}^T e_{k+1} \\
&= r_{k+1}^C + \bar{\zeta}_{k+1} V_{k+2} (\bar{\gamma}_{k+1} e_{k+1} - \beta_{k+2} c_k e_{k+2}) \quad \text{by (60) and (67)} \\
&= r_{k+1}^C + \bar{\zeta}_{k+1} (\bar{\gamma}_{k+1} v_{k+1} - \beta_{k+2} c_k v_{k+2}) \\
&= \bar{\zeta}_{k+1} \bar{\gamma}_{k+1} v_{k+1} - \\
&\quad \{ \beta_{k+2} (s_k \zeta_k - c_k \bar{\zeta}_{k+1}) + \bar{\zeta}_{k+1} \beta_{k+2} c_k \} v_{k+2} \quad \text{by (68)} \\
&= \zeta_{k+1} \gamma_{k+1} v_{k+1} - \epsilon_{k+2} \zeta_k v_{k+2} \quad \text{by (63) and (64)} \\
\Rightarrow \|r_k^L\|^2 &= \zeta_{k+1}^2 \gamma_{k+1}^2 + \epsilon_{k+2}^2 \zeta_k^2.
\end{aligned}$$

Because the implementation of SYMMLQ was originally obtained from this sort of approach in [PS75], there is no need to show any equivalence of this with a standard "vector" form here.

4.3 Minimum Residual Method (MINRES)

Problem: $\min_{x \in R^n} \|b - Ax\|$ for A symmetric indefinite, where A can be singular, and where $Ax = b$ need not be a consistent set of equations.

Process: the normalized Lanczos process for symmetric A on $\mathcal{K}^k(A; b)$ starting with $v_1 = b/\beta_1$, $\beta_1 \equiv \|b\|$.

Iterates of the form $x_k = V_k y_k$ are considered. Using (18),(19),(20) and (48),

$$\begin{aligned} r_k &= b - AV_k y_k = V_{k+1}(\beta_1 e_1 - T_{k+1,k} y_k) \\ \|r_k\| &= \|\beta_1 e_1 - T_{k+1,k} y_k\|. \end{aligned}$$

Subproblem: $\min_{y \in R^k} \|\beta_1 e_1 - T_{k+1,k} y_k\|$, that is in theory solve the normal equations $(T_k^2 + \beta_{k+1}^2 e_k e_k^T) y_k^M = \beta_1 T_k e_1$, for T_k symmetric tridiagonal. (The superscript M indicates the iterates $x_k^M = V_k y_k^M$ are of the MINRES method.) The matrix $T_{k+1,k}$ has full column rank, so the subproblem always has a unique solution.

Implementation: As for SYMMLQ, consider an LQ decomposition of T_k.

$$\begin{aligned} T_k \bar{Q}_k^T &= \bar{L}_k, \quad \bar{Q}_k^T \bar{Q}_k = I \\ T_{k+1,k}^T T_{k+1,k} &= T_k^2 + \beta_{k+1}^2 e_k e_k^T = T_k T_k^T + \beta_{k+1}^2 e_k e_k^T \\ &= \bar{L}_k \bar{L}_k^T + \beta_{k+1}^2 e_k e_k^T = L_k L_k^T. \end{aligned}$$

The last equality follows by considering the (k,k)-element of the matrices and (62). So the subproblem becomes:

$$\begin{aligned} L_k L_k^T y_k^M &= \beta_1 \bar{L}_k \bar{Q}_k e_1 \\ \text{or} \quad L_k^T y_k^M &= \beta_1 D_k \bar{Q}_k e_1, \end{aligned}$$

since L_k is nonsingular if $\beta_{k+1} \neq 0$, and from (62)

$$\bar{L}_k = L_k D_k \quad \text{where} \quad D_k \equiv \text{diag}(1, \ldots, 1, c_k).$$

For ease of computation define

$$t_k \equiv \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_k \end{bmatrix} \equiv \beta_1 D_k \bar{Q}_k e_1, \quad M_k \equiv \begin{bmatrix} m_1 & m_2 & \dots & m_k \end{bmatrix} \equiv V_k L_k^{-T}, \quad (69)$$

which transforms the system to

$$L_k^T y_k^M = t_k, \quad x_k^M = V_k y_k^M = V_k L_k^{-T} L_k^T y_k^M = M_k t_k.$$

By continuing the sequence

$$\bar{Q}_2 e_1 = Q_{1,2} e_1 = \begin{bmatrix} c_1 \\ s_1 \end{bmatrix}, \quad \bar{Q}_3 e_1 = Q_{2,3} Q_{1,3} e_1 = \begin{bmatrix} c_1 \\ c_2 s_1 \\ s_2 s_1 \end{bmatrix},$$

we see the entries of t_k are given by:

$$\tau_1 \equiv \beta_1 c_1, \quad \tau_i \equiv \beta_1 s_1 \cdots s_{i-1} c_i, \quad i = 2, \dots k. \quad (70)$$

The columns of M_k satisfy a three term recurrence ($M_k L_k^T = V_k$) (see (60)):

$$\begin{bmatrix} M_{k-1} & m_k \end{bmatrix} \left[\begin{array}{ccc|c} & & & 0 \\ & L_{k-1}^T & & \epsilon_k \\ & & & \delta_k \\ \hline 0 & \cdots & 0 & \gamma_k \end{array}\right] = \begin{bmatrix} & \epsilon_k m_{k-2} + \\ M_{k-1} L_{k-1}^T & \delta_k m_{k-1} + \\ & \gamma_k m_k \end{bmatrix}$$

$$\Rightarrow \quad m_1 = v_1/\gamma_1, \quad m_2 = (v_2 - \delta_2 m_1)/\gamma_2$$

$$m_k = (v_k - \delta_k m_{k-1} - \epsilon_k m_{k-2})/\gamma_k, \quad k > 2.$$

Finally:

$$x_k^M = M_k t_k = x_{k-1}^M + \tau_k m_k.$$

One may also consider the Conjugate Gradient solution, x_k^c.

$$\begin{aligned}
x_k^c &= V_k y_k^c \\
\text{where } T_k y_k^c &= \beta_1 e_1 \\
\bar{Q}_k T_k y_k^c &= \bar{L}_k^T y_k^c = \bar{Q}_k \beta_1 e_1 \\
x_k^c &= V_k L_k^{-T} L_k^T y_k^c = M_k L_k^T y_k^c \\
&= M_k (\bar{L}_k D_k^{-1})^T y_k^c = M_k D_k^{-1} \bar{L}_k^T y_k^c \\
&= M_k D_k^{-1} \bar{Q}_k \beta_1 e_1 = M_k D_k^{-2} (D_k \bar{Q}_k \beta_1 e_1) \\
&= M_k D_k^{-2} t_k, \qquad \text{by (69)} \\
&= x_k^M + \tau_k (c_k^{-2} - 1) m_k \\
&= x_k^M + \tau_k (s_k / c_k)^2 m_k.
\end{aligned}$$

So x_k^c is available from x_k^M, although the reverse does not appear to be true, that is, we have not derived a simple way of computing x_k^M from the Conjugate Gradient solutions method.

Residual Norm: To calculate $\|r_k^c\|$ or $\|r_k^M\|$, it is helpful to consider the relationship between the m_k, and the w_k as defined in (61) in SYMMLQ . Since we are considering the theoretical Lanczos process, it must stop in at most n steps. If it stops at $\beta_{m+1} = 0$, then

$$\begin{aligned}
AV_m &= V_m T_m, \quad T_m = L_m Q_m, \quad M_m = V_m L_m^{-T} \\
AM_m &= V_m T_m L_m^{-T} = V_m Q_m^T = W_m, \\
Am_k &= w_k, \quad k = 1, \ldots, m.
\end{aligned}$$

Now for the residuals

$$\begin{aligned}
r_k^M - r_k^c &\equiv A(x_k^c - x_k^M) = \tau_k (s_k / c_k)^2 w_k \\
r_k^M &\equiv b - A x_k^M = r_k^c + \tau_k (s_k / c_k)^2 w_k \\
&= \beta_1 s_1 \cdots s_k (s_k w_k - v_{k+1}) / c_k \quad \text{by (70)}
\end{aligned}$$

$$
\begin{aligned}
&= \beta_1 s_1 \cdots s_k \bar{w}_{k+1} \quad \text{by (65)} \\
\|r_k^M\| &= \beta_1 |s_1 \cdots s_k| \\
&= |c_k| \|r_k^c\|.
\end{aligned}
$$

As with SYMMLQ, the implementation of MINRES was originally obtained from this sort of approach in [PS75]. Thus there is no need to show any equivalence of this with a standard "vector" form here.

4.4 Conjugate Gradients on the Normal Equations (CGNE) – LSQR implementation

Problem: Solve $Ax = b$ if $A \in R^{m \times n}$ is consistent, otherwise $\min_{x \in R^n} \|b - Ax\|$ for general A, by (theoretically) performing the Conjugate Gradient algorithm on the Normal Equations[2],

$$A^T A x = A^T b, \tag{71}$$

which *is* a SPD system when A has rank n.

Process: the normalized Lanczos process for symmetric matrices applied to give a basis for $\mathcal{K}^k(A^T A; A^T b)$.

The practical implementation does not use $A^T A$ to compute the Lanczos vectors. Instead it considers two bidiagonalization procedures for the unsymmetric A called Bidiag1 and Bidiag2 respectively.

Bidiag1: This reduces A to lower bidiagonal form using

$$\begin{aligned} \beta_1 u_1 &= b \\ \alpha_1 v_1 &= A^T u_1 \\ \beta_{k+1} u_{k+1} &= A v_k - \alpha_k u_k \qquad (72) \\ \alpha_{k+1} v_{k+1} &= A^T u_{k+1} - \beta_{k+1} v_k \qquad (73) \end{aligned}$$

where the α_k and β_k are chosen to normalize v_k and u_k respectively, so $\|u_k\| = \|v_k\| = 1$. The process stops when either (72) or (73) is zero. With the definitions

$$U_k \equiv \begin{bmatrix} u_1 & u_2 & \dots & u_k \end{bmatrix}, \quad V_k \equiv \begin{bmatrix} v_1 & v_2 & \dots & v_k \end{bmatrix}$$

[2]See Claim at the end of this subsection, p. 51.

$$B_k \equiv \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \cdot & \\ & & \cdot & \alpha_k \end{bmatrix}, \quad B_{k+1,k} \equiv \begin{bmatrix} B_k \\ \beta_{k+1} e_k^T \end{bmatrix},$$

the recurrence relations may be written as

$$\begin{aligned} U_{k+1}(\beta_1 e_1) &= b \\ AV_k &= U_{k+1} B_{k+1,k} \qquad (74) \\ A^T U_{k+1} &= V_{k+1} B_{k+1}^T. \qquad (75) \end{aligned}$$

The beauty of this is that the choice of coefficients ensures in theory (prior to stopping)

$$U_{k+1}^T U_{k+1} = I = V_{k+1}^T V_{k+1}, \quad U_{k+1}^T A V_{k+1} = B_{k+1}.$$

Proof by induction:

$$\begin{aligned} B_1 &= \alpha_1 = v_1^T A^T u_1 = U_1^T A V_1 \\ U_1^T U_1 &= I = V_1^T V_1. \\ \text{Suppose } U_k^T U_k &= I = V_k^T V_k, \quad U_k^T A V_k = B_k. \\ \text{From (72) } \beta_{k+1} U_k^T u_{k+1} &= U_k^T A v_k - \alpha_k e_k = B_k e_k - \alpha_k e_k = 0, \\ \text{so } U_{k+1}^T U_{k+1} &= I \\ \text{and from (74) } U_{k+1}^T A V_k &= B_{k+1,k}. \\ \text{From (73) } \alpha_{k+1} V_k^T v_{k+1} &= V_k^T A^T u_{k+1} - \beta_{k+1} e_k \\ &= B_{k+1,k}^T e_k - B_{k+1,k}^T e_k = 0, \\ \text{so } V_{k+1}^T V_{k+1} &= I \\ \text{and from (75) } U_{k+1}^T A V_{k+1} &= B_{k+1}. \end{aligned}$$

Bidiag1 is related to two Lanczos processes, as can be seen by combining (74) and (75).

$$AA^T U_k = AV_k B_k^T = U_{k+1} B_{k+1,k} B_k^T,$$

$$A^TAV_k = A^TU_{k+1}B_{k+1,k} = V_{k+1}B_{k+1}^TB_{k+1,k}. \tag{76}$$

Clearly $\{u_i\}_1^k$ spans the krylov subspace $\mathcal{K}^k(AA^T;b)$, while $\{v_i\}_1^k$ spans the krylov subspace $\mathcal{K}^k(A^TA;A^Tb)$.

Bidiag2: This reduces A to upper bidiagonal form using

$$\begin{aligned} \theta_1 v_1 &= A^Tb \\ \rho_1 p_1 &= Av_1 \\ \theta_{k+1}v_{k+1} &= A^Tp_k - \rho_k v_k \\ \rho_{k+1}p_{k+1} &= Av_{k+1} - \theta_{k+1}p_k \end{aligned} \tag{77}$$

where the θ_k and ρ_k are chosen to give v_k and p_k unit norm respectively. With the definitions

$$P_k \equiv \begin{bmatrix} p_1 & p_2 & \dots & p_k \end{bmatrix}, \quad V_k \equiv \begin{bmatrix} v_1 & v_2 & \dots & v_k \end{bmatrix}$$

$$R_k \equiv \begin{bmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \cdot & \cdot & \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \end{bmatrix}, \quad R_{k,k+1} \equiv \begin{bmatrix} R_k \,\Big|\, \theta_{k+1}e_k \end{bmatrix},$$

the recurrence relations may be written as

$$\begin{aligned} V_k(\theta_1 e_1) &= A^Tb \\ A^TP_k &= V_kR_k^T + \theta_{k+1}v_{k+1}e_k^T = V_{k+1}R_{k,k+1}^T \\ AV_{k+1} &= P_{k+1}R_{k+1} \\ P_{k+1}^TP_{k+1} &= V_{k+1}^TV_{k+1} = I \\ P_{k+1}^TAV_{k+1} &= R_{k+1}. \end{aligned}$$

The last two lines follow since this recurrence is essentially Bidiag1 with b replaced by A^Tb, and A^T replaced by A. Again we have two Lanczos

processes

$$AA^TP_k = P_{k+1}R_{k+1}R_{k,k+1}^T, \qquad A^TAV_k = V_{k+1}R_{k,k+1}^TR_k, \tag{78}$$

so $\{p_i\}_1^k$ spans $\mathcal{K}^k(AA^T; AA^Tb)$, and $\{v_i\}_1^k$ spans $\mathcal{K}^k(A^TA; A^Tb)$.

Relationship between Bidiag1 and Bidiag2: The V_k are the same in Bidiag1 and Bidiag2, that is they are the result of applying the Lanczos process to develop $\mathcal{K}^k(A^TA; A^Tb)$. This follows since $\{v_i\}_1^k$ spans this same Krylov subspace in Bidiag1 and Bidiag2. The Lanczos processes with A^TA in (76) and (78) are therefore identical, and

$$B_{k+1}^TB_{k+1,k} = R_{k,k+1}^TR_k.$$

Dropping the last row of this gives

$$B_{k+1,k}^TB_{k+1,k} = R_k^TR_k.$$

This equality shows that R_k is the upper bidiagonal factor of $B_{k+1,k}$ in its QR factorisation.

$$\begin{aligned} Q_k^TB_{k+1,k} &= \begin{bmatrix} R_k \\ 0 \end{bmatrix} \quad \text{for } Q_k \text{ as in § 3,} \\ g_k &\equiv \beta_1Q_k^Te_1 \quad \text{as in (45).} \end{aligned}$$

The problem could be solved using either Bidiag1 or Bidiag2. Because of its greater numerical reliability in certain circumstances, LSQR was based on Bidiag1, and it is given here. It carries out the above QR factorization of $B_{k+1,k}$ as the process progresses.

To solve $\min_{x\in R^n} \|b - Ax\|$ using quantities from Bidiag1, let

$$\begin{aligned} x_k &= V_ky_k \\ t_{k+1} &\equiv \beta_1e_1 - B_{k+1,k}y_k \\ r_k &\equiv b - Ax_k \end{aligned}$$

$$
\begin{aligned}
&= U_{k+1}\beta_1 e_1 - U_{k+1}B_{k+1,k}y_k = U_{k+1}t_{k+1} \quad \text{by (74)} \\
\|r_k\| &= \|U_{k+1}t_{k+1}\| = \|t_{k+1}\| \\
\Rightarrow \min_{x\in\mathcal{K}^k} \|r_k\| &= \min_{y_k\in R^k} \|t_{k+1}\|
\end{aligned}
$$

Subproblem: $\min_{y_k\in R^k} \|\beta_1 e_1 - B_{k+1,k}y_k\|$, for $B_{k+1,k} \in \mathcal{R}^{(k+1)\times k}$ lower bidiagonal supplemented by an extra row. This always has a solution.

Implementation: This is solved using the QR factorisation of $B_{k+1,k}$ which with (45) transforms the subproblem to

$$
\begin{aligned}
&\min_{y_k\in R^k} \|Q_k^T\beta_1 e_1 - Q_k^T B_{k+1,k}y_k\| \\
&= \min_{y_k\in R^k} \|g_k - \begin{bmatrix} R_k \\ 0 \end{bmatrix} y_k\| \\
&= \min_{y_k\in R^k} \| \begin{bmatrix} f_k \\ \bar{\phi}_{k+1} \end{bmatrix} - \begin{bmatrix} R_k \\ 0 \end{bmatrix} y_k\|.
\end{aligned}
$$

This is equivalent to

$$
\begin{aligned}
R_k y_k &= f_k \\
t_{k+1} &= \beta_1 e_1 - Q_k \begin{bmatrix} R_k \\ 0 \end{bmatrix} y_k = Q_k g_k - Q_k \begin{bmatrix} f_k \\ 0 \end{bmatrix} \\
&= Q_k \begin{bmatrix} 0 \\ \bar{\phi}_{k+1} \end{bmatrix}.
\end{aligned}
$$

For ease of calculation define

$$
\begin{aligned}
D_k &\equiv \begin{bmatrix} d_1 & d_2 & \dots & d_k \end{bmatrix} = V_k R_k^{-1} \\
W_k &\equiv \begin{bmatrix} w_1 & w_2 & \dots & w_k \end{bmatrix} = D_k \mathrm{diag}(\rho_1, \dots, \rho_k) \\
x_k &= V_k y_k = V_k R_k^{-1} f_k = D_k f_k.
\end{aligned}
$$

Now the d_k (and their scalar multiples, the w_k) are easily calculated from

$V_k^T = R_k^T D_k^T$:

$$\begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix} = \begin{bmatrix} \rho_1 & & & \\ \theta_2 & \cdot & & \\ & \cdot & \rho_{k-1} & \\ & & \theta_k & \rho_k \end{bmatrix} \begin{bmatrix} d_1^T \\ d_2^T \\ \vdots \\ d_k^T \end{bmatrix}$$

$$\begin{aligned} (d_0 &\equiv 0, \quad w_0 \equiv 0) \\ d_k &= (v_k - \theta_k d_{k-1})/\rho_k \\ w_k &= v_k - \theta_k w_{k-1}/\rho_{k-1} \end{aligned}$$

and hence the x_k:

$$x_k = D_k f_k = D_{k-1} f_{k-1} + \phi_k d_k = x_{k-1} + \phi_k d_k.$$

Residual Norm:

$$\begin{aligned} r_k &= U_{k+1} t_{k+1} = U_{k+1} Q_k \begin{bmatrix} 0 \\ \bar{\phi}_{k+1} \end{bmatrix} = \bar{\phi}_{k+1} U_{k+1} Q_k e_{k+1} \\ \|r_k\| &= |\bar{\phi}_{k+1}| = \beta_1 |s_1 \cdots s_k| \quad \text{by (46)} \\ \Rightarrow \|r_k\| &= |s_k| \|r_{k-1}\|. \end{aligned}$$

Claim: LSQR is equivalent to CG on the Normal Eqautions (NE).

Proof: By $\hat{\ }$ denote the quantities from CG on the NE, that is

$$\hat{A}x = \hat{b},$$

where $\hat{A} \equiv A^T A$ is SPD and $\hat{b} \equiv A^T b$. Now it has already been shown that V_k produced in Bidiag1 is the same as V_k produced in the Lanczos process on $\mathcal{K}^k(\hat{A}, \hat{b})$. To show the theoretical equivalence of the two methods, we must prove

$$\begin{aligned} \hat{x}_k &= V_k \hat{y}_k \quad \text{is equivalent to} \quad x_k = V_k y_k \\ \text{or} \quad \hat{y}_k &= y_k. \end{aligned}$$

Now the y_k satisfy $R_k y_k = f_k$, while the $\hat{y}_k$ are obtained from

$$\begin{aligned} \hat{T}_k \hat{y}_k &= \hat{\beta}_1 e_1^{(k)} \\ R_k^T R_k \hat{y}_k &= \theta_1 e_1^{(k)} \quad \text{by (77) and (78)} \\ R_k \hat{y}_k &= R_k^{-T} \theta_1 e_1^{(k)}. \end{aligned}$$

So it remains to prove the equivalence of $\hat{y}_k$ and y_k using

$$\begin{aligned} R_k \hat{y}_k &= R_k^{-T} \theta_1 e_1^{(k)} \text{ and } R_k y_k = f_k. \\ \text{Now } R_k^T f_k &= \text{first } k \text{ entries of } \left[\begin{array}{c|c} R_k^T & \\ \hline & 0 \end{array}\right] g_k. \\ \left[\begin{array}{c|c} R_k^T & \\ \hline & 0 \end{array}\right] g_k &= \left[\begin{array}{c|c} R_k^T & \\ \hline & 0 \end{array}\right] \beta_1 Q_k^T e_1^{(k+1)} \\ &= \left(Q_k \left[\begin{array}{c|c} R_k & \\ \hline & 0 \end{array}\right] \right)^T \beta_1 e_1^{(k+1)} \\ &= \left[\begin{array}{c} B_{k+1,k}^T \\ \hline 0 \end{array}\right] \beta_1 e_1^{(k+1)} \\ &= \alpha_1 \beta_1 e_1^{(k+1)} = \theta_1 e_1^{(k+1)} \\ \Rightarrow f_k &= R_k^{-T} \theta_1 e_1^{(k)}. \end{aligned}$$

The two methods produce the same iterates and therefore are the same. However naively applying the CG method to the Normal Equations has unsatisfactory behaviour on ill-conditioned systems. Thus it is more favourable to use LSQR.

The LSQR implementation of CGNE was obtained from this sort of approach in [PS82]. The mathematical equivalence of LSQR with CGNE is shown above.

References: [GK65, PS82]

4.5 Generalized Minimum Residual Method with restarts (GMRES(m))

Problem: Solve $Ax = b$ for x when A is general nonsingular.

Process: Arnoldi algorithm applied to $\mathcal{K}^k(A; r_0)$, where $r_0 \equiv b - Ax_0$ for some initial guess x_0. Define $h_{1,0} \equiv \|r_0\|$, $v_1 = r_0/h_{1,0}$.

This algorithm considers iterates of the form $x_k = x_0 + V_k y_k$ so that $\|r_k\|$ is minimized over $(x_k - x_0) \in \mathcal{K}^k$, that is

$$\begin{aligned} \min_{x_k - x_0 \in \mathcal{K}^k} \|b - Ax_k\| &= \min_{y_k \in R^k} \|b - A(x_0 + V_k y_k)\| = \\ \min_{y_k \in R^k} \|r_0 - AV_k y_k\| &= \min_{y_k \in R^k} \|V_{k+1}(h_{1,0}e_1 - H_{k+1,k}y_k)\| \quad \text{by (3).} \end{aligned}$$

Subproblem: $\min_{y_k \in R^k} \|h_{1,0}e_1 - H_{k+1,k}y_k\|$, for $H_{k+1} \in R^{(k+1)\times k}$ upper Hessenberg supplemented by an extra row. This clearly always has a solution.

Implementation: This may be solved by performing a QR factorisation on $H_{k+1,k}$

$$Q_k^T H_{k+1,k} = \begin{bmatrix} R_k \\ 0 \end{bmatrix}, \quad g_k \equiv h_{1,0} Q_k^T e_1 = \begin{bmatrix} f_k \\ \bar{\phi}_{k+1} \end{bmatrix} \quad \text{as in (45).}$$

So by solving for $y_k = R_k^{-1} f_k$ by backward substitution, the subproblem becomes

$$\min_{y_k \in R^k} \|g_k - \begin{bmatrix} R_k y_k \\ 0 \end{bmatrix}\| = |\bar{\phi}_{k+1}|.$$

Residual Norm:

$$\begin{aligned} \|r_k\| &= |\bar{\phi}_{k+1}| = h_{1,0}|s_1 \cdots s_k| \quad \text{by (46)} \\ \Rightarrow \|r_k\| &= |s_k|\|r_{k-1}\|. \end{aligned}$$

As a practical note, since the $v_1, v_2, \ldots$ are kept, it is not necessary to compute x_k until iteration m, at which time

$$\begin{aligned} x_m &= V_m y_m \\ r_m &= b - Ax_m \end{aligned}$$

are calculated and the algorithm is restarted with $x_0 \equiv x_m$ and $r_0 \equiv r_m$. At any iteration, based on the magnitude of $\|r_k\|$, the algorithm may be stopped and x_k calculated.

The drawback of GMRES(n) requiring an increasing amount of work and storage as the iteration count rises is overcome by restarting every m iterations, for some $m \ll n$. Unfortunately the optimal value of m is difficult to determine and in certain cases the method will not converge for $m < n$. Too large a value of m results in unneccessary work and storage. On the other hand, too small a value may give slow convergence or fail to converge at all.

Because the implementation of GMRES was originally obtained from this sort of approach in [SS86], there is no need to show any equivalence of this with a standard "vector" form here. The GMRES(m) method — slightly altered from 'Templates' — is as follows.

x_0 initial guess
for $j = 1, 2, \ldots$
  $h_{1,0} = \|b - Ax_0\|$
  $v_1 = (b - Ax_0)/h_{1,0}$
  $Q_0^T = [1]$
  $\tilde{\phi}_1 = h_{1,0}$
  for $i = 1, 2, \ldots, m$
    $u_i = Av_i$
    for $k = 1, 2, \ldots, i$
      $h_{k,i} = v_k^T u_i$ (or, k-th entry h_i gets $h_{k,i}$)
      $u_i = u_i - h_{k,i} v_k$
    $h_{i+1,i} = \|u_i\|$
    $v_{i+1} = u_i / h_{i+1,i}$

$$R_i = \left[\begin{array}{c|c} R_{i-1} & Q_{i-1}^T h_i \\ 0 & \end{array}\right]$$

$\tilde{r}_{i,i} = r_{i,i}$

$r_{i,i} = \sqrt{\tilde{r}_{i,i}^2 + h_{i+1,i}^2}$

$c_i = \tilde{r}_{i,i}/r_{i,i}, \quad s_i = -h_{i+1,i}/r_{i,i}$

$$Q_i^T = Q_{i,i}^T \left[\begin{array}{c|c} Q_{i-1}^T & \\ \hline & 1 \end{array}\right]$$

$\phi_i = c_i \bar{\phi}_i$

$$f_i = \left[\begin{array}{c} f_{i-1} \\ \phi_i \end{array}\right]$$

$\bar{\phi}_{i+1} = s_i \bar{\phi}_i$

if $|\bar{\phi}_{i+1}| <$ tol

update(x, i) and quit

update(x, m)

end

update(x, j):

solve $R_j y_j = f_j$ by back substitution

form $x_j = V_j y_j$

end

4.6 The Unsymmetric Lanczos (BiConjugate Gradients) Method

Problem: Solve $Ax = b$ for x when A is general nonsingular.

First we derive what will later be shown to be the BiConjugate Gradients approximations x_k from the standard unsymmetric Lanczos process. We refer to this as:

The standard unsymmetric Lanczos-BiCG method.

Process: the standard unsymmetric Lanczos process (32) to produce mutually orthogonal $\{v_1, \ldots, v_k\}$ and $\{\tilde{v}_1, \ldots, \tilde{v}_k\}$ spanning the krylov subspaces $\mathcal{K}^k(A; b)$ and $\mathcal{K}^k(A^T; b)$ respectively, starting with $v_1 = b/\beta_1$ and $\tilde{v}_1 = b/\gamma_1$, where $\beta_1 = \gamma_1 = \|b\|$, where β_{j+1} is chosen to normalize v_{j+1}.

To derive the subproblem we use (31) and consider iterates of the form $x_k = V_k y_k$ as suggested in (47), and take $T_k y_k = \beta_1 e_1$, so that

$$\begin{aligned} r_k &= b - Ax_k = b - AV_k y_k \\ &= b - V_k T_k y_k - \beta_{k+1} v_{k+1} e_k^T y_k \\ &= V_k(\beta_1 e_1 - T_k y_k) - \beta_{k+1} v_{k+1} e_k^T y_k \\ &= -\beta_{k+1} v_{k+1} e_k^T y_k, \end{aligned} \tag{79}$$

much the same as (49) in the unit norm Lanczos method. In theory we have $\beta_{k+1} v_{k+1} = 0$ for some $k \leq n$, the dimension of A, and the problem will be solved.

Subproblem: Solve $T_k y_k = \beta_1 e_1$, where T_k is tridiagonal.

When T_k is singular this subproblem does not have a (unique) solution, but as we are only intent on giving a new approach to the general BiCG idea, we do not go into such aspects here. One very satisfactory approach to avoiding such difficulties is to use the Quasi-Minimal Residuals method discussed later in § 4.7 . This work on BiCG will motivate that.

Implementation: This may be solved using an LU factorisation (when it exists) of $T_k = L_k U_k$, where L_k is lower bidiagonal and U_k is chosen to be unit upper bidiagonal. There is nothing special about T_k, so the use of this decomposition without pivoting may not be stable. For further insights on this, see the following alternative unsymmetric Lanczos-BiCG method.

For ease of calculation and savings on storage, define

$$z_k \equiv \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_k \end{bmatrix} \equiv U_k y_k, \quad W_k \equiv \begin{bmatrix} w_1 & w_2 & \cdots & w_k \end{bmatrix} \equiv V_k U_k^{-1},$$

so the system is transformed to

$$T_k y_k = L_k U_k y_k = L_k z_k \;\Rightarrow\; L_k z_k = \beta_1 e_1,$$
$$V_k y_k = V_k U_k^{-1} U_k y_k \;\Rightarrow\; x_k = W_k z_k.$$

Now, L_k is formed one row at a time; U_k one column at a time:

$$T_k = \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1} & 0 \\ \hline 0 & \epsilon_k & \delta_k \end{array}\right] \left[\begin{array}{cc|c} & & 0 \\ & U_{k-1} & \eta_k \\ \hline 0 & 0 & 1 \end{array}\right]$$

$$= \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1}U_{k-1} & \delta_{k-1}\eta_k \\ \hline 0 & \epsilon_k & \epsilon_k\eta_k + \delta_k \end{array}\right] = \left[\begin{array}{cc|c} & & 0 \\ & T_{k-1} & \gamma_k \\ \hline 0 & \beta_k & \alpha_k \end{array}\right]$$

$$\Rightarrow \delta_1 = \alpha_1$$

$$\epsilon_k = \beta_k, \quad \eta_k = \gamma_k/\delta_{k-1}, \quad \delta_k = \alpha_k - \epsilon_k\eta_k, \quad k > 1. \qquad (80)$$

And so z_k is formed one entry at a time:

$$\left[\begin{array}{cc|c} & & 0 \\ & L_{k-1} & 0 \\ \hline 0 & \epsilon_k & \delta_k \end{array}\right] \begin{bmatrix} z_{k-1} \\ \zeta_k \end{bmatrix} = \begin{bmatrix} L_{k-1}z_{k-1} \\ \epsilon_k\zeta_{k-1} + \delta_k\zeta_k \end{bmatrix} = \beta_1 e_1$$

$$\Rightarrow \zeta_1 = \beta_1/\delta_1, \quad \zeta_k = -\epsilon_k\zeta_{k-1}/\delta_k, \quad k > 1. \qquad (81)$$

W_k is also formed one column at a time ($W_k U_k = V_k$):

$$\left[\begin{array}{cc} W_{k-1} & w_k \end{array}\right] \left[\begin{array}{cc|c} & & 0 \\ & U_{k-1} & \\ & & \eta_k \\ \hline 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cc} W_{k-1}U_{k-1} & \eta_k w_{k-1} + w_k \end{array}\right]$$

$$\Rightarrow w_1 = v_1, \quad w_k = v_k - \eta_k w_{k-1}, \quad k > 1.$$

Finally,

$$x_k = W_k z_k = W_{k-1} z_{k-1} + \zeta_k w_k = x_{k-1} + \zeta_k w_k, \quad k \geq 1.$$

Residual Norm: The following estimation of the norm of the residual r_k, as defined in (79) can be used.

$$\begin{aligned} r_k &= -\beta_{k+1} v_{k+1} e_k^T y_k = -\beta_{k+1} v_{k+1} y_{kk} \\ \|r_k\| &= \beta_{k+1} |y_{kk}|, \quad \text{where } y_{kk} \text{ is the } k\text{-th entry of } y_k. \end{aligned}$$

Since y_k is not computed directly, one must consider

$$\begin{aligned} U_k y_k &= z_k \\ \left[\begin{array}{cc|c} & & 0 \\ & U_{k-1} & \\ & & \eta_k \\ \hline 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} y_{k1} \\ \vdots \\ y_{kk} \end{array}\right] &= \left[\begin{array}{c} \vdots \\ y_{kk} \end{array}\right] = \left[\begin{array}{c} z_{k-1} \\ \zeta_k \end{array}\right] \\ y_{kk} &= \zeta_k \\ \Rightarrow \|r_k\| &= \beta_{k+1} |\zeta_k| \\ &= \beta_{k+1} |\epsilon_k \zeta_{k-1} / \delta_k| \quad \text{by (81)} \\ &= \beta_{k+1} \beta_k |\zeta_{k-1} / \delta_k| \quad \text{by (80)} \\ \|r_k\| &= \beta_{k+1} \|r_{k-1}\| / |\delta_k|, \quad k \geq 1. \end{aligned}$$

Although this solves the problem in much the same way as BiCG in 'Templates', [BBC+94], it gives different scalings for equivalent vectors. Let us now derive a method which, as with the BiCG method of [BBC+94], produces the residual vectors themselves.

The alternative unsymmetric Lanczos-BiCG method

Process: the alternative unsymmetric Lanczos process, see (33) to (44), starting with $v_1 = \tilde{v}_1 = b$ $(\beta_1 \equiv 1)$. The normalization β_{j+1} is unspecified as yet.

Using (37) and and considering iterates of the form $x_k = V_k y_k$ as suggested in (47) and solving for $T_k y_k = e_1$,

$$\begin{aligned} r_k &= b - Ax_k = b - AV_k y_k \\ &= b - V_k T_k y_k - \beta_{k+1} v_{k+1} e_k^T y_k \\ &= V_k(e_1 - T_k y_k) - \beta_{k+1} v_{k+1} e_k^T y_k \\ &= -\beta_{k+1} v_{k+1} e_k^T y_k. \end{aligned} \tag{82}$$

Subproblem: Solve $T_k y_k = e_1$, where T_k is tridiagonal. Again T_k may be singular, in which case this approach, and ordinary BiCG, fails.

Implementation: This may be solved using an LDU decomposition of T_k. The one shown below suits our purposes. With the as yet unkown $\mathcal{D}_k \equiv -\mathrm{diag}(\beta_2, \ldots, \beta_{k+1})$, write

$$\begin{aligned} T_k &= L_k \mathcal{D}_k U_k \\ &= \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1} & 0 \\ \hline 0 & -1 & 1 \end{array}\right] \left[\begin{array}{cc|c} & & 0 \\ & \mathcal{D}_{k-1} & 0 \\ \hline 0 & 0 & -\beta_{k+1} \end{array}\right] \left[\begin{array}{cc|c} & & 0 \\ & U_{k-1} & -\frac{\gamma_k}{\beta_k} \\ \hline 0 & 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{cc|c} & & 0 \\ & L_{k-1}\mathcal{D}_{k-1}U_{k-1} & \\ & & \gamma_k \\ \hline 0 & \beta_k & -(\gamma_k + \beta_{k+1}) \end{array}\right] \end{aligned} \tag{83}$$

A simpler method could be obtained by combining $\mathcal{D}$ and U, and using for example $\mathcal{U} \equiv \mathcal{D}U$ instead, but the approach here will produce the equivalents of various elements of BiCG that we want.

Our normalizers β_{j+1} may now be determined by equating the right side

of the above equation with the tridiagonal T_k in (36). So

$$\begin{aligned} \alpha_1 &= -\beta_2; \quad \alpha_k = -(\gamma_k + \beta_{k+1}), \quad k > 1, \\ \Rightarrow \beta_2 &\equiv -\alpha_1; \quad \beta_{k+1} \equiv -(\alpha_k + \gamma_k), \quad k > 1. \end{aligned} \tag{84}$$

We will show later that if $\beta_{j+1} = 0$ the ordinary BiCG breaks down too, otherwise this is a satisfactory normalizer for the process, and U_{k+1} exists. In the meantime define

$$z_k \equiv \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_k \end{bmatrix} \equiv U_k y_k, \quad W_k \equiv \begin{bmatrix} w_1 & w_2 & \cdots & w_k \end{bmatrix} \equiv V_k U_k^{-1}. \tag{85}$$

For our later analysis, we also define

$$\widetilde{W}_k \equiv \begin{bmatrix} \tilde{w}_1 & \tilde{w}_2 & \cdots & \tilde{w}_k \end{bmatrix} \equiv \widetilde{V}_k U_k^{-1},$$

although this is not used in the algorithm itself.

The system is transformed to

$$\begin{aligned} T_k y_k = L_k \mathcal{D}_k U_k y_k = L_k \mathcal{D}_k z_k \quad &\Rightarrow \quad L_k \mathcal{D}_k z_k = e_1 \\ V_k y_k = V_k U_k^{-1} U_k y_k \quad &\Rightarrow \quad x_k = W_k z_k. \end{aligned}$$

Now z_k is formed one entry at a time:

$$\begin{aligned} L_k \mathcal{D}_k z_k &= \begin{bmatrix} -\beta_2 \zeta_1 \\ \beta_2 \zeta_1 - \beta_3 \zeta_2 \\ \vdots \\ \beta_k \zeta_{k-1} - \beta_{k+1} \zeta_k \end{bmatrix} = e_1 \\ \Rightarrow \zeta_1 &= -\frac{1}{\beta_2} \\ \zeta_k &= \frac{\beta_k}{\beta_{k+1}} \zeta_{k-1} = \frac{\beta_{k-1}}{\beta_{k+1}} \zeta_{k-2} = \cdots = -\frac{1}{\beta_{k+1}}, \quad k > 1. \end{aligned} \tag{86}$$

Solving $W_k U_k = V_k$ and $\widetilde{W}_k U_k = \tilde{V}_k$,

$$\begin{aligned} w_1 &= v_1, \quad w_k = v_k + \frac{\gamma_k}{\beta_k} w_{k-1}, \quad k > 1 \\ \tilde{w}_1 &= \tilde{v}_1, \quad \tilde{w}_k = \tilde{v}_k + \frac{\gamma_k}{\beta_k} \tilde{w}_{k-1}, \quad k > 1. \end{aligned}$$

Finally,

$$x_k = W_{k-1} z_{k-1} + \zeta_k w_k = x_{k-1} + \zeta_k w_k.$$

Residual Norm: The following estimation of the norm of the residual r_k, as defined in (S2) may be used.

$$\begin{aligned} r_k &= -\beta_{k+1} y_{kk} v_{k+1} \\ &= -\beta_{k+1} \zeta_k v_{k+1} \quad \text{by (S5)} \\ &= v_{k+1} \quad \text{by (S6)} \qquad (S7) \\ \Rightarrow \|r_k\| &= \|v_{k+1}\|. \end{aligned}$$

So the alternative unsymmetric Lanczos process, with $v_1 = b$, can be used to produce the (BiCG) residual vectors $r_k = b - Ax_k$ themselves, as long as β_{j+1} is chosen as in (S4).

It remains to examine under what conditions β_{j+1} can be zero. Suppose $\beta_2, \ldots, \beta_j$ obtained from (S4) are nonzero. Then with $e^T \equiv [1, \ldots, 1]$, we have $e^T T_j = [0, \ldots, 0, \alpha_j + \gamma_j]$, which will be nonzero if T_j is nonsingular. However if $\alpha_j + \gamma_j = 0$ then T_j is singular, and both this and the standard BiCG method fails. Otherwise, it is clear that the $L_k U_k$ 'factorization' is possible. However small β_k could cause large element growth in U_k.

Equivalence of unsymmetric Lanczos-BiCG with BiCG: The BiCG method — slightly altered from 'Templates' — is as follows.

$r_0 = b$

$\tilde{r}_0 = r_0$

for $i = 1, 2, \ldots$

$\rho_{i-1} = r_{i-1}^T \tilde{r}_{i-1}$

if $\rho_{i-1} = 0$, the method fails, stop

if $(i = 1)$

$p_i = r_{i-1}$

$\tilde{p}_i = \tilde{r}_{i-1}$

else

$\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$

$p_i = r_{i-1} + \beta_{i-1} p_{i-1}$

$\tilde{p}_i = \tilde{r}_{i-1} + \beta_{i-1} \tilde{p}_{i-1}$

$q_i = A p_i$

$\tilde{q}_i = A^T \tilde{p}_i$

$\alpha_i = \rho_{i-1}/\tilde{p}_i^T q_i$

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

$r_i = r_{i-1} - \alpha_i q_i$

$\tilde{r}_i = \tilde{r}_{i-1} - \alpha_i \tilde{q}_i$

check convergence; continue if necessary

end

To avoid confusion we will use the superscript ^{B} to distinguish 'Templates' terms from those here, where it will also make life easier if we re-index two of the 'Templates' terms as follows:

$$\rho_i^B \equiv \text{'Templates'}\rho_{i-1},$$
$$\beta_i^B \equiv \text{'Templates'}\beta_{i-1}.$$

We now write the algorithms side by side for comparison, where in the alternative unsymmetric Lanczos-BiCG algorithm we replace v_{i+1} by r_i, see (57), introduce $u_i \equiv A v_i$ and $\tilde{u}_i \equiv A^T \tilde{v}_i$ and redefine $\tilde{r}_i \equiv \tilde{v}_{i+1}$. For ease of comparison in handling the $i = 1$ case we assume $1/\infty = 0$ and define

BiCG	unsymmetric Lanczos-BiCG
$\rho_0^B = 1$	$\rho_0 = \infty, \quad \beta_1 = 1$
$x_0^B = 0, \quad p_0^B = 0, \quad \tilde{p}_0^B = 0$	$x_0 = 0$
$\tilde{r}_0^B = r_0^B = b$	$r_{-1} = 0 \quad \tilde{r}_{-1} = 0, \tilde{r}_0 = r_0 = b$

Then for $i = 1, 2, 3, \ldots$ we have

BiCG	alt. unsym. Lanczos-BiCG
$\rho_i^B = r_{i-1}^{BT} \tilde{r}_{i-1}^B$	$\rho_i = \tilde{r}_{i-1}^T r_{i-1}$
$\beta_i^B = \rho_i^B / \rho_{i-1}^B$	$\gamma_i = \beta_i \rho_i / \rho_{i-1}$ from (44)
$p_i^B = r_{i-1}^B + \beta_i^B p_{i-1}^B$	$w_i = r_{i-1} + (\gamma_i/\beta_i) w_{i-1}$
$\tilde{p}_i^B = \tilde{r}_{i-1}^B + \beta_i^B \tilde{p}_{i-1}^B$	$\tilde{w}_i = \tilde{r}_{i-1} + (\gamma_i/\beta_i) \tilde{w}_{i-1}$
$q_i^B = A p_i^B$	$u_i = A r_{i-1}$
$\tilde{q}_i = A^T \tilde{p}_i^B$	$\tilde{u}_i = A^T \tilde{r}_{i-1}$
$\alpha_i^B = \rho_i^B / \tilde{p}_i^{BT} q_i^B$	$\alpha_i = \tilde{r}_{i-1}^T u_i / \rho_i$ from (40)
	$\beta_{i+1} = -(\alpha_i + \gamma_i)$
$x_i^B = x_{i-1}^B + \alpha_i^B p_i^B$	$x_i = x_{i-1} - (1/\beta_{i+1}) w_i$
$r_i^B = r_{i-1}^B - \alpha_i^B q_i^B$	$r_i = (u_i - \alpha_i r_{i-1} - \gamma_i r_{i-2})/\beta_{i+1}$
$\tilde{r}_i^B = \tilde{r}_{i-1}^B - \alpha_i^B \tilde{q}_i^B$	$\tilde{r}_i = (\tilde{u}_i - \alpha_i \tilde{r}_{i-1} - \gamma_i \tilde{r}_{i-2})/\beta_{i+1}$

The main difference in cost between the two is in the calculation of the residuals and shadow residuals (the $\tilde{r}_i$) — $2n$ extra multiplications, subtractions and divisions in the alternative unsymmetric Lanczos-BiCG algorithm — which are partially offset by the n extra multiplications and additions in the BiCG calculation of $\tilde{p}_i^B$ ($\tilde{w}_i$ is not actually calculated). The equivalences are

$$\begin{aligned} \rho_i &= \rho_i^B, \quad \gamma_i/\beta_i = \beta_i^B, \quad w_i = p_i^B, \quad \tilde{w}_i = \tilde{p}_i^B \\ -1/\beta_{i+1} &= \alpha_i^B, \quad x_i = x_i^B, \quad r_i = r_i^B, \quad \tilde{r}_i = \tilde{r}_i^B. \end{aligned} \tag{88}$$

A straightforward examination shows this is true for $i = 1$. Suppose it is

true for $i = 1, 2, \ldots, k-1$. Then it is clear that

$$\rho_k = \rho_k^B, \quad \gamma_k/\beta_k = \beta_k^B, \quad w_k = p_k^B, \quad \tilde{w}_k = \tilde{p}_k^B.$$

We now need only prove

$$-1/\beta_{k+1} = \alpha_k^B, \tag{89}$$

for then $x_k = x_k^B$, and their residuals r_k and r_k^B must then also be equal, since r_i^B is clearly the residual for x_i^B by examining the last two lines of BiCG, and we have shown r_i is the residual of x_i in the alternative unsymmetric Lanczos-BiCG. In this case $\tilde{r}_k = \tilde{r}_k^B$ follows from the equality of the r_k and r_k^B residuals, since we know that $r_k = \varphi_k(A)b$ and $r_k^B = \psi_k(A)b$ for two k degree polynomials φ_k and ψ_k, and $\varphi_k = \psi_k$. But we can also show $\tilde{r}_k = \varphi_k(A^T)b$ and $\tilde{r}_k^B = \psi_k(A^T)b$, so these must also be equal. We can prove (89) using our matrix formulation of the unsymmetric Lanczos-BiCG algorithm. We know $W_k = V_k U_k^{-1}$ and $\widetilde{W}_k = \tilde{V}_k U_k^{-1}$, so

$$\begin{aligned} \widetilde{W}_k^T A W_k &= U_k^{-T} \tilde{V}_k^T A V_k U_k^{-1} = U_k^{-T} D_k T_k U_k^{-1} \\ &= U_k^{-T} D_k (L_k \mathcal{D}_k U_k) U_k^{-1} = U_k^{-T} D_k L_k \mathcal{D}_k, \end{aligned}$$

which, by the way, is lower triangular and symmetric (see (39)), and therefore diagonal, showing "A-biorthogonality" of the w_i and $\tilde{w}_j$, where $i, j < k$. The (i,i)-element of this is (see (83))

$$\tilde{w}_i^T A w_i = e_i^T U_k^{-T} (\rho_i e_i - \rho_{i+1} e_{i+1})(-\beta_{i+1}).$$

But $e_i^T U_k^{-T} = (\times, \ldots, \times, 1, 0, \ldots, 0)$ is zero after the i-th element, so since $p_i^B = w_i$ $\tilde{p}_i^B = \tilde{w}_i$, $i = 1, \ldots, k$,

$$\tilde{p}_i^{BT} A p_i^B = \tilde{w}_i^T A w_i = -\rho_i \beta_{i+1}, \qquad i = 1, \ldots, k.$$

From the BiCG algorithm we then see

$$\alpha_i^B = \rho_i^B / \tilde{p}_i^{BT} q_i^B = \rho_i / \tilde{p}_i^{BT} A p_i^B = -1/\beta_{i+1}, \quad i = 1, \ldots, k,$$

and so (89) holds, and the induction proof of (88) is complete, showing that in theory these algorithms not only compute the same iterates, but also several of the intermediate quantities are identical. However the algorithms are clearly not identical, as the last two lines of each clearly highlights, and may have different numerical behaviours. This suggests a matrix development of BiCG would be useful, but we leave this for future work.

4.7 Quasi-Minimal Residuals (QMR)

Problem: Solve $Ax = b$ for x when A is general nonsingular.

Process: an unsymmetric Lanczos process which produces mutually orthonormal $\{v_1, \ldots, v_k\}$ and $\{\tilde{v}_1, \ldots, \tilde{v}_k\}$ spanning the krylov spaces $\mathcal{K}^k(A; b)$ and $\mathcal{K}^k(A^T; b)$ respectively, starting with $v_1 = b/\beta_1$ and $\tilde{v}_1 = b/\tilde{\beta}_1$, where $\beta_1 = \tilde{\beta}_1 = \|b\|$. We choose β_{j+1} to make v_{j+1} unit length and $\tilde{\beta}_{j+1}$ to make $\tilde{v}_{j+1}$ unit length. We define $\rho_{j+1} \equiv \tilde{v}_{j+1}^T v_{j+1}$.

To derive the subproblem we use (21) (where $T_k \equiv H_k$ is tridiagonal) and consider iterates of the form $x_k = V_k y_k$ as suggested in (47).

$$r_k = b - Ax_k = \beta_1 v_1 - AV_k y_k = V_{k+1}(\beta_1 e_1 - T_{k+1,k} y_k). \tag{90}$$

Since V_{k+1} does not have orthonormal columns, it is difficult to minimize $\|r_k\|$. The reduced system below is considered, and this gives rise to the name 'Quasi', since the residual is not actually minimised.

From (23) $\tilde{V}_{k+1}^T V_{k+1} = D_{k+1}$, so $\tilde{V}_{k+1}^T r_k = D_{k+1}(\beta_1 e_1 - T_{k+1,k} y_k)$. QMR minimizes $\|\beta_1 e_1 - T_{k+1,k} y_k\|$, so in reality (in theory) QMR is minimizing $\|D_{k+1}^{-1} \tilde{V}_{i+1}^T r_k\|$, not $\|r_k\|$.

Subproblem: $\min_{y_k \in R^k} \|\beta_1 e_1 - T_{k+1,k} y_k\|$, for $T_{k+1,k} \in \mathcal{R}^{(k+1)\times k}$ tridiagonal supplemented by an extra row. This always has a solution.

Implementation: This may be solved using a QR factorisation of $T_{k+1,k}$ as was done on $H_{k+1,k}$ in GMRES in § 4.5.

$$Q_k^T T_{k+1,k} = \begin{bmatrix} R_k \\ 0 \end{bmatrix}, \quad g_k \equiv \beta_1 Q_k^T e_1 = \begin{bmatrix} f_k \\ \bar{\phi}_{k+1} \end{bmatrix}.$$

However R_k is upper tridiagonal, not just upper triangular. Solving for $y_k = R_k^{-1} f_k$ the subproblem becomes

$$\min_{y_k \in R^k} \|g_k - \begin{bmatrix} R_k \\ 0 \end{bmatrix} y_k\| = |\bar{\phi}_{k+1}|. \tag{91}$$

For ease of calculation define

$$W_k \equiv \left[\begin{array}{cccc} w_1 & w_2 & \cdots & w_k \end{array} \right] \equiv V_k R_k^{-1}.$$

The w_k satisfy a three term recurrence ($W_k R_k = V_k$):

$$\left[\begin{array}{cc} W_{k-1} & w_k \end{array} \right] \left[\begin{array}{c|c} R_{k-1} & \begin{array}{c} 0 \\ r_{k-2,k} \\ r_{k-1,k} \end{array} \\ \hline 0 \;\; 0 \;\; 0 & r_{kk} \end{array} \right] = \left[\begin{array}{cc} W_{k-1}R_{k-1} & v_k \end{array} \right]$$

$$\Rightarrow \quad w_1 = v_1/r_{1,1}, \quad w_2 = (v_2 - r_{1,2}w_1)/r_{2,2}$$

$$w_k = (v_k - r_{k-2,k}w_{k-2} - r_{k-1,k}w_{k-1})/r_{k,k}, \quad k > 2.$$

Finally,

$$\begin{aligned} x_k &= V_k y_k = V_k R_k^{-1} f_k = W_k f_k \\ &= W_{k-1} f_{k-1} + \phi_k w_k = x_{k-1} + \phi_k w_k. \end{aligned}$$

Residual and Residual Norm: The following estimation of the norm of the residual r_k, as defined in (90) is used.

$$\begin{aligned} \|r_k\| &= \|V_{k+1}(\beta_1 e_1 - T_{k+1,k} y_k)\| \\ &\leq \sqrt{k+1}\|\beta_1 e_1 - T_{k+1,k} y_k\| \quad \text{(columns of } V_{k+1} \text{ are unit length)} \\ &= \sqrt{k+1}|\bar{\phi}_{k+1}| \quad \text{by (91)} \\ &= \beta_1\sqrt{k+1}|s_1 \cdots s_k| \quad \text{by (46).} \end{aligned}$$

Once this bound on the residual norm reaches a sufficently low value, the following calculation of the actual residual is used for the few remaining iterations.

$$\begin{aligned} r_k &= V_{k+1}(\beta_1 e_1 - T_{k+1,k} R_k^{-1} f_k) \\ &= \bar{\phi}_{k+1} V_{k+1} Q_k e_{k+1} \end{aligned} \tag{92}$$

$$
\begin{aligned}
&= \bar{\phi}_{k+1} V_{k+1} \left[\begin{array}{c|c} Q_{k-1} & \\ \hline & 1 \end{array}\right] (s_k e_k + c_k e_{k+1}) \\
&= \bar{\phi}_{k+1} (s_k V_k Q_{k-1} e_k + c_k v_{k+1}) \\
&= s_k^2 \bar{\phi}_k V_k Q_{k-1} e_k + c_k \bar{\phi}_{k+1} v_{k+1} \quad \text{by (46)} \\
&= s_k^2 r_{k-1} + c_k \bar{\phi}_{k+1} v_{k+1} \quad \text{by (92).}
\end{aligned}
$$

Finally, here is the unpreconditioned algorithm with no look-ahead or scaling from the original paper [FN91].

$x_0 = 0$

$\beta_1 = \|b\|, \quad \tilde{\beta}_1 = \beta_1$

$v_1 = b/\beta_1, \quad \tilde{v}_1 = v_1$

$\rho_1 = \tilde{v}_1^T v_1$

$u_1 = A v_1, \quad \tilde{u}_1 = A^T \tilde{v}_1$

$c_0 = 1$

$\bar{\phi}_1 = \beta_1$

$r_{-1,1} w_{-1} = 0, \quad r_{0,1} w_0 = 0$

for $j = 1, 2, \ldots$

$\alpha_j = \tilde{v}^T u_j / \rho_j$

$v_{j+1} = u_j - \alpha_j v_j$

$\beta_{j+1} = \|v_{j+1}\|$

$v_{j+1} = v_{j+1} / \beta_{j+1}$

$\tilde{v}_{j+1} = \tilde{u}_j - \alpha_j \tilde{v}_j$

$\tilde{\beta}_{j+1} = \|\tilde{v}_{j+1}\|$

$\tilde{v}_{j+1} = \tilde{v}_{j+1} / \tilde{\beta}_{j+1}$

$\rho_{j+1} = \tilde{v}_{j+1}^T v_{j+1}$

$\gamma_{j+1} = \tilde{\beta}_{j+1} \rho_{j+1} / \rho_j$

$u_{j+1} = A v_{j+1} - \gamma_{j+1} v_j$

$\tilde{\gamma}_{j+1} = \beta_{j+1} \rho_{j+1} / \rho_j$

$\tilde{u}_{j+1} = A^T \tilde{v}_{j+1} - \tilde{\gamma}_{j+1} \tilde{v}_j$

if $j > 2$

$r_{j-2,j} = -\gamma_j s_{j-2}$

if $j > 1$

$r_{j-1,j} = \gamma_j c_{j-1} c_{j-2} - \alpha_j s_{j-1}$

$\tilde{r}_{j,j} = \gamma_j s_{j-1} c_{j-2} + \alpha_j c_{j-1}$

$r_{j,j} = \sqrt{\tilde{r}_{j,j}^2 + \beta_{j+1}^2}$

$c_j = \tilde{r}_{j,j} / r_{j,j} \quad s_j = -\beta_{j+1} / r_{j,j}$

$\phi_j = c_j \bar{\phi}_j, \quad \bar{\phi}_{j+1} = s_j \bar{\phi}_j$

$w_j = (v_j - r_{j-2,j} w_{j-2} - r_{j-1,j} w_{j-1}) / r_{j,j}$

$x_j = x_{j-1} + \phi_j w_j$

check convergence; continue if necessary

end

4.8 Conjugate Gradients Squared (CGS)

Motivation

The Conjugate Gradients Squared (CGS) method was so named because in a sense (that will become clear) it gives the "square" of some quantities in the BiCG method, and the best introduction to it appears to be via a new CGS algorithm derived from the alternative unsymmetric Lanczos-BiCG algorithm of § 4.6, which we reproduce here for ease of reference. First the initial conditions are (where again we use $1/\infty = 0$):

$$\rho_0 = \infty, \quad \beta_1 = 1, \quad x_0 = 0, \quad r_{-1} = 0 = \tilde{r}_{-1} = 0, \quad r_0 = \tilde{r}_0 = b.$$

Then for $i = 1, 2, 3, \ldots$ we have

alternative unsymmetric Lanczos-BiCG

$\rho_i = \tilde{r}_{i-1}^T r_{i-1}$

$\gamma_i = \beta_i \rho_i / \rho_{i-1}$ from (44)

$w_i = r_{i-1} + (\gamma_i/\beta_i) w_{i-1}$

$\tilde{w}_i = \tilde{r}_{i-1} + (\gamma_i/\beta_i) \tilde{w}_{i-1}$

$u_i = A r_{i-1}$

$\tilde{u}_i = A^T \tilde{r}_{i-1}$

$\alpha_i = \tilde{r}_{i-1}^T u_i / \rho_i$ from (40)

$\beta_{i+1} = -(\alpha_i + \gamma_i)$

$x_i = x_{i-1} - (1/\beta_{i+1}) w_i$

$r_i = (u_i - \alpha_i r_{i-1} - \gamma_i r_{i-2})/\beta_{i+1}$

$\tilde{r}_i = (\tilde{u}_i - \alpha_i \tilde{r}_{i-1} - \gamma_i \tilde{r}_{i-2})/\beta_{i+1}$

We see from the initial conditions and last two lines that

$$r_0 \equiv b,$$

$$r_k = \varphi_k(A) r_0, \quad \tilde{r}_k = \varphi_k(A^T) r_0, \quad k \geq 0,$$

where the polynomial in A, $\varphi_k(A)$, expressed as a scalar polynomial $\varphi_k \equiv \varphi_k(\theta)$ is given by

$$\begin{aligned} \varphi_0 &\equiv 1, \quad \varphi_1 = \frac{1}{\beta_2}(\theta - \alpha_1)\varphi_0, \\ \varphi_k &= \frac{1}{\beta_{k+1}}[(\theta - \alpha_k)\varphi_{k-1} - \gamma_k\varphi_{k-2}], \quad k > 1. \end{aligned} \tag{93}$$

Now since the BiCG residuals are constructed to be mutually orthogonal,

$$\begin{aligned} \tilde{r}_i^T r_j &= r_0^T \varphi_i(A)\varphi_j(A)r_0 = 0 \quad \text{for} \quad i \neq j, \\ &= r_0^T \varphi_i^2(A)r_0 \neq 0 \quad \text{for} \quad i = j. \end{aligned}$$

It is the squared polynomial in this last line which is important in CGS. Notice in BiCG that multiplication by A and A^T was required in going from r_{i-1} to r_i; that is from $\varphi_{i-1}(A)r_0$ to $\varphi_i(A)r_0$. Using CGS we will see that it is possible to go from $\varphi_{i-1}^2(A)r_0$ to $\varphi_i^2(A)r_0$ using only two multiplications by A. The advantages of this are that multiplications by A^T are not required (and sometimes it is cumbersome to produce code to do such multiplications), and $2i$ matrix multiplications are required to produce a polynomial of degree $2i$ (whereas it is only degree i in BiCG). A disadvantage is that although $\varphi_i(A)r_0$ has useful properties in BiCG, it is not clear what the useful properties of $\varphi_i^2(A)r_0$ *are*. Nevertheless we now derive a new variant of CGS. Define

$$\begin{aligned} \tilde{r}^{CGS} &\equiv r_0, \\ r_k^{CGS} &\equiv \varphi_k^2(A)r_0, \quad k \geq 0, \\ \text{and} \quad s_k^{CGS} &\equiv \varphi_k(A)\varphi_{k-1}(A)r_0, \quad k \geq 1. \\ \text{Then} \quad (\tilde{r}^{CGS})^T s_k^{CGS} &= r_0^T \varphi_k(A)\varphi_{k-1}(A)r_0 \\ &= \tilde{r}_k^T r_{k-1} = 0, \quad k \geq 1. \end{aligned} \tag{94}$$

Now we show how these new polynomials φ_k^2 and $\varphi_k\varphi_{k-1}$ may be constructed

recursively, using (93). From the resulting expressions we give the corresponding recurrences for r_k^{CGS} and s_k^{CGS}.

$$\begin{aligned}
\varphi_1\varphi_0 &= \varphi_1 = \frac{1}{\beta_2}(\theta - \alpha_1), \qquad \text{or} \quad s_1^{CGS} = \varphi_1 = \frac{1}{\beta_2}(A - \alpha_1 I)r_0, \\
\varphi_k\varphi_{k-1} &= \frac{1}{\beta_{k+1}}[(\theta - \alpha_k)\varphi_{k-1}^2 - \gamma_k\varphi_{k-1}\varphi_{k-2}], \quad k > 1, \\
s_k^{CGS} &= \frac{1}{\beta_{k+1}}[(A - \alpha_k I)r_{k-1}^{CGS} - \gamma_k s_{k-1}^{CGS}], \quad k > 1. \qquad (95) \\
\varphi_1^2 &= \frac{1}{\beta_2}(\theta - \alpha_1)\varphi_1\varphi_0, \quad \text{or} \quad r_1^{CGS} = \frac{1}{\beta_2}(A - \alpha_1 I)s_1^{CGS}, \qquad (96)
\end{aligned}$$

and for $k > 1$

$$\begin{aligned}
\varphi_k^2 &= \frac{1}{\beta_{k+1}^2}[(\theta - \alpha_k)^2\varphi_{k-1}^2 - 2\gamma_k(\theta - \alpha_k)\varphi_{k-1}\varphi_{k-2} + \gamma_k^2\varphi_{k-2}^2], \\
&= \frac{1}{\beta_{k+1}^2}[\beta_{k+1}(\theta - \alpha_k)\varphi_k\varphi_{k-1} - \gamma_k(\theta - \alpha_k)\varphi_{k-1}\varphi_{k-2} + \gamma_k^2\varphi_{k-2}^2], \\
r_k^{CGS} &= \frac{1}{\beta_{k+1}^2}[\beta_{k+1}(A - \alpha_k I)s_k^{CGS} - \gamma_k(A - \alpha_k I)s_{k-1}^{CGS} + \gamma_k^2 r_{k-2}^{CGS}]. \quad (97)
\end{aligned}$$

The key to this new CGS algorithm is that the α_k, γ_k and β_{k+1} are exactly those in the Lanczos-BiCG algorithm above, and may be computed from the CGS vectors as follows.

$$\begin{aligned}
\rho_i &= \tilde{r}_{i-1}^T r_{i-1} = \tilde{r}_0^T\varphi_{i-1}^2(A)r_0 = (\tilde{r}^{CGS})^T r_{i-1}^{CGS}, \quad i \geq 1, \qquad (98) \\
\gamma_1 &= 0, \quad \gamma_i = \beta_i\rho_i/\rho_{i-1}, \quad i > 1, \\
\alpha_i &= \tilde{r}_{i-1}^T A r_{i-1}/\rho_i = \tilde{r}_0^T\varphi_{i-1}(A)A\varphi_{i-1}(A)r_0/\rho_i \\
&= (\tilde{r}^{CGS})^T A\varphi_{i-1}^2(A)r_0/\rho_i = (\tilde{r}^{CGS})^T A r_{i-1}^{CGS}/\rho_i, \quad i \geq 1 \\
&= (\tilde{r}^{CGS})^T(A r_{i-1}^{CGS} - \gamma_i s_{i-1}^{CGS})/\rho_i, \quad i \geq 2, \qquad (99)
\end{aligned}$$

since $(\tilde{r}^{CGS})^T s_{i-1}^{CGS} = 0$ for $i \geq 2$. Finally $\beta_2 = -\alpha_1$ and $\beta_{i+1} = -(\alpha_i + \gamma_i)$, for $i > 1$, are immediately available.

We can now combine the above expressions to give the full computation for the r_k^{CGS} and s_k^{CGS} vectors — we call this the "CGS process" – without yet showing the computation for the solution, which is given later.

New CGS method

Problem: Solve $Ax = b$ for x when A is general nonsingular.

CGS Process: We drop the superscript CGS from s_j, and write $\tilde{v} \equiv \tilde{r}^{CGS} = r_0$; $v_{j+1} \equiv r_j^{CGS}$, $j \geq 0$, and for efficient computation define $\hat{u}_1 \equiv Av_1$; $\hat{u}_j \equiv Av_j - \gamma_j s_{j-1}$ for $j > 1$, and $t_j \equiv As_j$ for $j \geq 1$. Again, $1/\infty = 0$.

$$\begin{aligned}
\rho_0 &\equiv \infty, \quad \beta_1 = 1, \quad s_0 \equiv 0, \quad v_0 \equiv 0 \\
v_1 &= \tilde{v} = b \\
\text{for} \quad j &= 1, 2, \ldots \\
&\rho_j = \tilde{v}^T v_j \quad \text{from (98)} \\
&\gamma_j = \frac{\rho_j}{\rho_{j-1}} \beta_j \\
&\hat{u}_j = Av_j - \gamma_j s_{j-1} \\
&\alpha_j = \tilde{v}^T \hat{u}_j / \rho_j \quad \text{from (99)} \\
&\beta_{j+1} = -(\alpha_j + \gamma_j) \qquad (100) \\
&\beta_{j+1} s_j = \hat{u}_j - \alpha_j v_j \quad \text{from (95)} \qquad (101) \\
&t_j = As_j \\
&\beta_{j+1} v_{j+1} = t_j - \alpha_j s_j - \frac{\gamma_j}{\beta_{j+1}} (t_{j-1} - \alpha_j s_{j-1}) \\
&\qquad + \frac{\gamma_j^2}{\beta_{j+1}} v_{j-1} \qquad (102)
\end{aligned}$$

this last coming from (97). A healthy aspect of this is that, as required in (94),

$$\beta_{j+1} \tilde{v}^T s_j = \tilde{v}^T \hat{u}_j - \alpha_j \tilde{v}^T v_j = \alpha_j \rho_j - \alpha_j \rho_j = 0, \quad j \geq 1, \qquad (103)$$

follows immediately from the jth step without any recourse to additional theory or induction.

Despite the elegance of the above motivation for the new CGS method and process, the reader should be warned that the following is essentially

"work in progress", and the matrix representation is somewhat complicated, and probably not optimal.

Matrix representation

If we use (101) for the odd columns and (102) for the even columns the CGS process becomes, in matrix form

$$\begin{aligned} \bar{V}_{2k} &\equiv \left[\begin{array}{cccccc} \bar{v}_1 & \bar{v}_2 & \cdots & \bar{v}_{2k-1} & \bar{v}_{2k} \end{array} \right] \qquad (104) \\ &\equiv \left[\begin{array}{cccccc} v_1 & s_1 & \cdots & v_k & s_k \end{array} \right] \\ A\bar{V}_{2k}G_{2k} &= \bar{V}_{2k}F_{2k} + \beta_{k+1}\bar{v}_{2k+1}e_{2k}^T = \bar{V}_{2k+1}F_{2k+1,2k} \qquad (105) \end{aligned}$$

$$\text{where} \quad G_2 \equiv \left[\begin{array}{cc} 1 & \\ & 1 \end{array} \right], \quad G_{2k} \equiv \left[\begin{array}{cc|cc} & & -\frac{\gamma_k}{\beta_{k+1}} & \\ \multicolumn{2}{c|}{\multirow{2}{*}{G_{2k-2}}} & & \\ & & 1 & \\ & & & 1 \end{array} \right], \quad k > 1$$

$$F_2 \equiv \left[\begin{array}{cc} \alpha_1 & \\ \beta_2 & \alpha_1 \end{array} \right],$$

$$F_{2k} \equiv \left[\begin{array}{cc|cc} & & & -\frac{\gamma_k^2}{\beta_{k+1}} \\ \multicolumn{2}{c|}{\multirow{2}{*}{F_{2k-2}}} & \gamma_k & -\frac{\alpha_k\gamma_k}{\beta_{k+1}} \\ \hline & \beta_k & \alpha_k & \\ & & \beta_{k+1} & \alpha_k \end{array} \right], \quad k > 1 \qquad (106)$$

$$F_{2k+1,2k} \equiv \left[\begin{array}{c} F_{2k} \\ \beta_{k+1}e_{2k}^T \end{array} \right].$$

Using (105) and solving for $F_{2k}\bar{y}_{2k} = e_1$, we consider iterates x_k not quite the same as in (47),

$$\begin{aligned} x_k &= \bar{x}_{2k} = \bar{V}_{2k}G_{2k}\bar{y}_{2k} \qquad (107) \\ \Rightarrow \quad r_k &= b - Ax_k = b - A\bar{x}_{2k} = b - A\bar{V}_{2k}G_{2k}\bar{y}_{2k} \\ &= b - (\bar{V}_{2k}F_{2k} + \beta_{k+1}\bar{v}_{2k+1}e_{2k}^T)\bar{y}_{2k} \end{aligned}$$

$$= \bar{V}_{2k}(e_1 - F_{2k}\bar{y}_{2k}) - \beta_{k+1}v_{k+1}e_{2k}^T\bar{y}_{2k}$$
$$= -\beta_{k+1}\bar{y}_{2k,2k}v_{k+1}. \qquad (108)$$

Subproblem: Solve $F_{2k}\bar{y}_{2k} = e_1$, where F_{2k} is upper Hessenberg as in (106). This does not have a solution if F_{2k} is singular, and at that time this implementation of the CGS method would fail. However, we continue assuming that this is not the case.

Implementation: This may be solved using an LDU decomposition of F_{2k}. The one which suits our purposes is shown below.

$$F_2 = L_2\bar{D}_2\bar{U}_2$$
$$= \begin{bmatrix} 1 & \\ -1 & 1 \end{bmatrix}\begin{bmatrix} -\beta_2 & \\ & -\beta_2 \end{bmatrix}\begin{bmatrix} 1 & \\ & 1 \end{bmatrix} = \begin{bmatrix} -\beta_2 & \\ \beta_2 & -\beta_2 \end{bmatrix}$$

$$F_{2k} = L_{2k}\bar{D}_{2k}\bar{U}_{2k}$$

$$L_{2k} = \left[\begin{array}{cc|cc} & & & \\ & L_{2k-2} & & \\ & & & \\ \hline & -1 & 1 & \\ & & -1 & 1 \end{array}\right], \quad \bar{D}_{2k} = \left[\begin{array}{cc|cc} & & & \\ & \bar{D}_{2k-2} & & \\ & & & \\ \hline & & -\beta_{k+1} & \\ & & & -\beta_{k+1} \end{array}\right]$$

$$\bar{U}_{2k} = \left[\begin{array}{cc|cc} & & & \frac{\gamma_k^2}{\beta_k\beta_{k+1}} \\ & \bar{U}_{2k-2} & & \\ & & & \\ \hline & & -\frac{\gamma_k}{\beta_k} & -\frac{\gamma_k}{\beta_k} \\ & & 1 & -\frac{\gamma_k}{\beta_{k+1}} \\ & & & 1 \end{array}\right]$$

$$\Rightarrow F_{2k} = \left[\begin{array}{cc|cc} & & & -\frac{\gamma_k^2}{\beta_{k+1}} \\ & L_{2k-2}\bar{D}_{2k-2}\bar{U}_{2k-2} & & \\ & & \gamma_k & \frac{\gamma_k}{\beta_{k+1}}(\gamma_k + \beta_{k+1}) \\ \hline & \beta_k & -(\gamma_k + \beta_{k+1}) & \\ & & \beta_{k+1} & -(\gamma_k + \beta_{k+1}) \end{array}\right].$$

Our normalizers β_{j+1} are now verified by equating the right sides of the

above equations with F_2 and F_{2k}. So

$$\begin{aligned} \alpha_1 &= -\beta_2; \quad \alpha_k = -(\gamma_k + \beta_{k+1}), \quad k > 1 \\ \Rightarrow \beta_2 &\equiv -\alpha_1; \quad \beta_{k+1} \equiv -(\alpha_k + \gamma_k), \quad k > 1. \end{aligned} \tag{109}$$

This was the choice of β_{j+1} that we made in § 4.6. As long as it is nonzero, it is a satisfactory normalizer for the process, and $\bar{U}_{2k+2}$ exists. For ease of calculation, we make the following definitions.

$$\begin{aligned} \bar{W}_{2k} &\equiv \begin{bmatrix} \bar{w}_1 & \bar{w}_2 & \cdots & \bar{w}_{2k-1} & \bar{w}_{2k} \end{bmatrix} \\ &\equiv \begin{bmatrix} u_1 & q_1 & \cdots & u_k & q_k \end{bmatrix} \equiv \bar{V}_{2k} G_{2k} \bar{U}_{2k}^{-1} \end{aligned}$$

$$\text{and} \quad \bar{z}_{2k} \equiv \begin{bmatrix} \bar{\zeta}_1 \\ \bar{\zeta}_2 \\ \vdots \\ \bar{\zeta}_{2k} \end{bmatrix} \equiv \bar{U}_{2k} \bar{y}_{2k}$$

$$\Rightarrow \bar{\zeta}_{2k} = \bar{y}_{2k,2k}.$$

So our system is transformed to

$$L_{2k} \bar{D}_{2k} \bar{z}_{2k} = e_1, \quad \bar{x}_{2k} = \bar{W}_{2k} \bar{z}_{2k}.$$

Solving

$$L_{2k} \bar{D}_{2k} \bar{z}_{2k} = \begin{bmatrix} -\beta_2 \bar{\zeta}_1 \\ \beta_2 \bar{\zeta}_1 - \beta_2 \bar{\zeta}_2 \\ \beta_2 \bar{\zeta}_2 - \beta_3 \bar{\zeta}_3 \\ \vdots \\ \beta_k \bar{\zeta}_{2k-2} - \beta_{k+1} \bar{\zeta}_{2k-1} \\ \beta_{k+1} \bar{\zeta}_{2k-1} - \beta_{k+1} \bar{\zeta}_{2k} \end{bmatrix} = e_1$$

$$\begin{aligned} \Rightarrow \quad \bar{\zeta}_1 &= -\frac{1}{\beta_2} \\ \zeta_1 \equiv \bar{\zeta}_2 &= \bar{\zeta}_1 = -\frac{1}{\beta_2} \end{aligned}$$

$$\begin{aligned}
\bar{\zeta}_{2k-1} &= \frac{\beta_k}{\beta_{k+1}}\bar{\zeta}_{2k-2}, \quad k>1 \\
\zeta_k &\equiv \bar{\zeta}_{2k} = \bar{\zeta}_{2k-1} = \frac{\beta_k}{\beta_{k+1}}\bar{\zeta}_{2k-2} \qquad (110)\\
&= \frac{\beta_{k-1}}{\beta_{k+1}}\bar{\zeta}_{2k-4} = \cdots = -\frac{1}{\beta_{k+1}}, \quad k>1.
\end{aligned}$$

Consider the odd columns of $\bar{W}_{2k}\bar{U}_{2k} = \bar{V}_{2k}G_{2k}$.

$$\begin{aligned}
\Rightarrow \quad u_1 &= \bar{w}_1 = \bar{v}_1 = v_1 \\
u_k &= \bar{w}_{2k-1} = \bar{v}_{2k-1} + \frac{\gamma_k}{\beta_k}\bar{w}_{2k-2} = v_k + \frac{\gamma_k}{\beta_k}q_{k-1}, \quad k>1. \quad (111)
\end{aligned}$$

The even columns give a less pleasing result.

$$\begin{aligned}
q_1 &= \bar{w}_2 = s_1 \\
q_k &= \bar{w}_{2k} = s_k - \frac{\gamma_k}{\beta_{k+1}}(s_{k-1}+u_k) + \frac{\gamma_k}{\beta_k}(q_{k-1} - \frac{\gamma_k}{\beta_{k+1}}u_{k-1}), \quad k>1.
\end{aligned}$$

$$\begin{aligned}
\text{Now} \quad r_k &= -\beta_{k+1}\bar{y}_{2k,2k}v_{k+1} = -\beta_{k+1}\bar{\zeta}_{2k}v_{k+1} = v_{k+1} \qquad (112)\\
\text{and} \quad x_k &= \bar{x}_{2k-2} + \bar{\zeta}_{2k-1}u_k + \bar{\zeta}_{2k}q_k = x_{k-1} - \frac{1}{\beta_{k+1}}(u_k+q_k) \quad (113)
\end{aligned}$$

so this CGS process produces vectors v_{j+1} which are the residual vectors r_j of the method.

It remains to examine under what conditions β_{j+1} can be zero. Supose $\beta_2,\ldots,\beta_j$ obtained from (109) are nonzero. The with $e^T \equiv [1,\ldots,1]$, we have

$$\begin{aligned}
e^T F_{2j-1} &= [0,\cdots,-\frac{\gamma_{j-1}^2}{\beta_j} - \frac{\alpha_{j-1}\gamma_{j-1}}{\beta_j} + \alpha_{j-1} + \beta_j, \alpha_j+\gamma_j] \\
e^T F_{2j-1}e_{2j-2} &= \frac{-\gamma_{j-1}^2 - \alpha_{j-1}\gamma_{j-1} - \alpha_{j-1}(\alpha_{j-1}+\gamma_{j-1}) + (\alpha_{j-1}+\gamma_{j-1})^2}{\beta_j} \\
&= 0 \\
\Rightarrow e^T F_{2j-1} &= [0,\cdots,0,\alpha_j+\gamma_j],
\end{aligned}$$

which willl be nonzero if F_{2j-1} is nonsingular. However if $\alpha_j+\gamma_j = 0$ then F_{2j-1} is singular, and both this and the standard CGS method fails.

Otherwise it is clear that the $L_{2k}\bar{D}_{2k}\bar{U}_{2k}$ 'factorization' is possible. However small β_k could cause large element growth in $\bar{U}_{2k}$.

Let us introduce vectors $P_k \equiv [p_1, p_2, \ldots, p_k]$, not used in the algorithm, but useful for this analysis.

$$P_k \equiv \bar{W}_{2k} M_{2k,k} E_k^{-1}$$

$$\text{where} \quad M_{2k,k} \equiv \left[\begin{array}{c|c} M_{2k-2,k-1} & \begin{matrix} \frac{\gamma_k}{\beta_k} \\ \hline 1 \\ 0 \end{matrix} \end{array}\right], \quad E_k \equiv \left[\begin{array}{c|c} E_{k-1} & \begin{matrix} -\frac{\gamma_k^2}{\beta_k^2} \\ \hline 1 \end{matrix} \end{array}\right].$$

$$\text{So} \quad p_1 = u_1$$

$$p_k = u_k + \frac{\gamma_k}{\beta_k}(q_{k-1} + \frac{\gamma_k}{\beta_k} p_{k-1}), \quad k > 1. \tag{114}$$

Equivalence of this method with the vector form:

The CGS method — slightly altered from 'Templates' -- is as follows.

$r_0 = b$

$\tilde{r} = r_0$

for $i = 1, 2, \ldots$

  $\rho_{i-1} = \tilde{r}^T r_{i-1}$

  if $\rho_{i-1} = 0$, method fails

  if $i = 1$

    $u_i = r_{i-1}$

    $p_i = u_i$

  else

    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$

    $u_i = r_{i-1} + \beta_{i-1} q_{i-1}$

    $p_i = u_i + \beta_{i-1}(q_{i-1} + \beta_{i-1} p_{i-1})$

$\hat{v} = Ap_i$

$\alpha_i = \rho_{i-1}/\tilde{r}^T\hat{v}$

$q_i = u_i - \alpha_i\hat{v}$

$\hat{u} = u_i + q_i$

$x_i = x_{i-1} + \alpha_i\hat{u}$

$\hat{q} = A\hat{u}$

$r_i = r_{i-1} - \alpha_i\hat{q}$

check convergence; continue if necessary

end

To avoid confusion we will use the superscript ^{C} to distinguish 'Templates' terms from those here, where it will also make life easier if we re-index two of the 'Templates' terms as follows:

$$\rho_i^C \equiv \text{'Templates'}\rho_{i-1},$$
$$\beta_i^C \equiv \text{'Templates'}\beta_{i-1}.$$

We now write the algorithms side by side for comparison, where in this new CGS algorithm we replace v_{i+1} by r_i, see (112). To handle the $i = 1$ case we define

CGS	new CGS
$\rho_0^C = 1$	$\rho_0 = \infty, \beta_1 = 1$
$p_0^C = 0, q_0^C = 0$	$u_0 = 0, p_0 = 0, s_0 = 0, q_0 = 0$

Then for $i = 1, 2, 3, \ldots$ we have

CGS	new CGS
$\rho_i^C = \tilde{r}^{CT} r_{i-1}^C$	$\rho_i = \tilde{r}^T r_{i-1}$
$\beta_i^C = \rho_i^C / \rho_{i-1}^C$	$\gamma_i = \beta_i \rho_i / \rho_{i-1}$
$u_i^C = r_{i-1}^C + \beta_i^C q_{i-1}^C$	$u_i = r_{i-1} + \frac{\gamma_i}{\beta_i} q_{i-1}$

$p_i^C = u_i^C + \beta_i^C(q_{i-1}^C + \beta_i^C p_{i-1}^C)$	$p_i = u_i + \frac{\gamma_i}{\beta_i}(q_{i-1} + \frac{\gamma_i}{\beta_i}p_{i-1})$
$\alpha_i^C = \rho_i^C / \tilde{r}^{CT} A p_i^C$	$\alpha_i = \tilde{r}^T(Ar_{i-1} - \gamma_i s_{i-1})/\rho_i$
	$\beta_{i+1} = -(\alpha_i + \gamma_i)$
	$s_i = (Ar_{i-1} - \alpha_i r_{i-1} - \gamma_i s_{i-1})/\beta_{i+1}$
$q_i^C = u_i^C - \alpha_i^C A p_i^C$	$q_i = s_i - \frac{\gamma_i}{\beta_{i+1}}(s_{i-1} + u_i)$
	$+\frac{\gamma_i}{\beta_i}(q_{i-1} - \frac{\gamma_i}{\beta_{i+1}}u_{i-1})$
$x_i^C = x_{i-1}^C + \alpha_i^C(u_i^C + q_i^C)$	$x_i = x_{i-1} - 1/\beta_{i+1}(u_i + q_i)$
$r_i^C = r_{i-1}^C - \alpha_i^C A(u_i^C + q_i^C)$	$r_i = \frac{1}{\beta_{i+1}}(As_i - \alpha_i s_i)$
	$-\frac{\gamma_i}{\beta_{i+1}^2}(As_{i-1} - \alpha_i s_{i-1}) + \frac{\gamma_i^2}{\beta_{i+1}^2}r_{i-2}$

There are several differences between the two algorithms — they don't even use all the same vectors — the new CGS uses the s_i which are orthogonal to $\tilde{r}$, while CGS produces the p_k^C. The other vectors are the same; the exact relationships are

$$\begin{aligned} \gamma_i/\beta_i &= \beta_i^C, \quad i > 1 \\ \rho_i &= \rho_i^C, \quad -1/\beta_{i+1} = \alpha_i^C \\ u_i &= u_i^C, \quad p_i = p_i^C \\ s_i &= r_{i-1}^C - \alpha_i^C A u_i^C, \quad q_i = q_i^C \\ x_i &= x_i^C, \quad r_i = r_i^C. \end{aligned} \tag{115}$$

A straightforward examination shows this is true for $i = 1$. Suppose it is true for $i = 1, 2, \ldots, k-1$. Then it is clear that

$$\rho_k = \rho_k^C, \quad \gamma_k/\beta_k = \beta_k^C, \quad u_k = u_k^C, \quad p_k = p_k^C.$$

We now need only prove

$$-1/\beta_{k+1} = \alpha_k^C \tag{116}$$

$$s_k = r_{k-1}^C - \alpha_k^C A u_k^C \tag{117}$$

$$q_k = q_k^C$$

so that $x_k = x_k^C$. The equality of r_k and r_k^C follows since we have shown that r_k is the residual of x_k and r_k^C is obviously the residual of x_k^C by examining the last two lines of CGS. We can prove (116) using our matrix formulation of the new CGS algorithm. Consider

$$\begin{aligned}
\tilde{v}^T A P_k &= \tilde{v}^T A \bar{W}_{2k} M_{2k,k} E_k^{-1} \\
&= \tilde{v}^T A \bar{V}_{2k} G_{2k} \bar{U}_{2k}^{-1} M_{2k,k} E_k^{-1} \\
&= \tilde{v}^T \bar{V}_{2k+1} F_{2k+1,2k} \bar{U}_{2k}^{-1} M_{2k,k} E_k^{-1} \\
&= \tilde{v}^T \bar{V}_{2k+1} L_{2k+1,2k} \bar{D}_{2k} M_{2k,k} E_k^{-1} \\
&= \begin{bmatrix} \rho_1 & 0 & \rho_2 & \cdots & 0 & \rho_{k+1} \end{bmatrix} L_{2k+1,2k} \bar{D}_{2k} M_{2k,k} E_k^{-1}.
\end{aligned}$$

The (i,i)- element of this is

$$\begin{aligned}
\tilde{v}^T A p_i &= \begin{bmatrix} -\rho_1\beta_2 & \rho_2(\gamma_2 - \beta_3) & \cdots & \rho_k(\gamma_k - \beta_{k+1}) \end{bmatrix} \begin{bmatrix} (\frac{\gamma_2 \cdots \gamma_i}{\beta_2 \cdots \beta_i})^2 \\ (\frac{\gamma_3 \cdots \gamma_i}{\beta_3 \cdots \beta_i})^2 \\ \vdots \\ (\frac{\gamma_i}{\beta_i})^2 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
&= -\rho_i \beta_{i+1}
\end{aligned}$$

after much cancellation. Now since $\tilde{r}^C = \tilde{v}$ and $p_i^C = p_i$, $i = 1, \ldots, k$,

$$\tilde{r}^{CT} A p_i^C = \tilde{v}^T A p_i = -\rho_i \beta_{i+1}, \quad i = 1, \ldots, k.$$

From the CGS algorithm we then see

$$\alpha_i^C = \rho_i^C / \tilde{r}^{CT} A p_i^C = \rho_i / \tilde{r}^{CT} A p_i^C = -1/\beta_{i+1}, \quad i = 1, \ldots, k,$$

and so (116) holds. Now, to show $s_k = r_{k-1}^C - \alpha_k^C A u_k^C = r_{k-1} + 1/\beta_{k+1} A u_k$, look at

$$A u_i = A \bar{W}_{2k} e_{2i-1}$$

$$
\begin{aligned}
&= \bar{V}_{2k+1} L_{2k+1,2k} \bar{D}_{2k} e_{2i-1} \\
&= -\beta_{i+1}(v_i - s_i).
\end{aligned}
$$

So s_i is alternatively (and more simply) given by the above expression and (117) falls out. Finally, $q_k = q_k^C$. For lack of a simpler way to show this, consider the matrix equation

$$
\begin{aligned}
AP_k &= A\bar{W}_{2k} M_{2k,k} E_k^{-1} \\
&= \bar{W}_{2k+1} \bar{U}_{2k+1} G_{2k+1}^{-1} L_{2k+1,2k} \bar{D}_{2k} M_{2k,k} E_k^{-1}.
\end{aligned}
$$

It is not obvious, but this has i-th column

$$
\begin{aligned}
Ap_i &= \bar{W}_{2k+1} \beta_{i+1} (e_{2i} - e_{2i-1}) \\
&= \beta_{i+1}(q_i - u_i), \quad i = 1, \ldots, k \\
\Rightarrow q_k &= u_k + 1/\beta_{k+1} Ap_k \\
&= u_k^C - \alpha_k^C A p_k^C.
\end{aligned}
$$

The induction proof (115) is complete, showing that in theory these algorithms not only compute the same iterates, but also several of the intermediate quantities are identical. However the algorithms are clearly not identical, as the use of different vectors shows, and may have different numerical behaviours. This suggests a matrix development of CGS would be useful, but we leave this for future work.

Reference: [Son89]

4.9 BiConjugate Gradients Stabilized (BiCGSTAB)

Motivation

The BiConjugate Gradients Stabilized (BiCGSTAB) method was so named because it increases the convergence of BiCG but in a more stable way than blindly "squaring" the residual vectors r_i as in CGS. Instead we form $r_i = \varphi_i(A)\theta_i(A)r_0$, where $\theta_i(\theta) \equiv (1-\omega_i\theta)\theta_{i-1}$ is an ith degree polynomial describing a steepest descent update — ω_i is chosen to minimize the norm of r_i — in the hope that these new iterates will converge more regularly than the $\varphi_i^2(A)r_0$ of CGS. Here is our new variant of BiCGSTAB. Using $\varphi_i \equiv \varphi_i(\theta)$ from § 4.8, define

$$\begin{aligned} \theta_0 \equiv 1, \quad \theta_k &= (1-\omega_k\theta)\theta_{k-1}, \quad k \geq 1, \qquad (113)\\ \tilde{r}^{BCS} &\equiv r_0, \\ r_k^{BCS} &\equiv \varphi_k(A)\theta_k(A)r_0, \quad k \geq 0, \\ s_k^{BCS} &\equiv \varphi_k(A)\theta_{k-1}(A)r_0, \quad k \geq 1, \\ \text{and} \quad u_k^{BCS} &\equiv \varphi_{k-1}(A)\theta_k(A)r_0, \quad k \geq 1. \\ \text{Then} \quad (\tilde{r}^{BCS})^T s_k^{BCS} &= r_0^T\varphi_k(A)\theta_{k-1}(A)r_0 \\ &= \tilde{r}_k^T(\theta_{k-1}(A)r_0) = 0 \qquad (119)\\ \text{since} \quad \tilde{r}_k &\perp \theta_{k-1}(A)r_0 \in \mathcal{K}^k(A;r_0). \end{aligned}$$

These new polynomials, $\varphi_{k-1}\theta_k, \varphi_k\theta_{k-1}$ and $\varphi_k\theta_k$, may be constructed recursively using (93) and (118). From these expressions we give the corresponding recurrences for r_k^{BCS}, s_k^{BCS} and u_k^{BCS}.

$$\begin{aligned} \varphi_{k-1}\theta_k &= (1-\omega_k\theta)\varphi_{k-1}\theta_{k-1}, \quad k \geq 1 \\ \text{or} \quad u_k^{BCS} &= (I-\omega_k A)r_{k-1}^{BCS}, \quad k \geq 1. \\ \varphi_1\theta_0 &= \frac{1}{\beta_2}(\theta-\alpha_1)\varphi_0\theta_0, \quad \text{or} \quad s_1^{BCS} = \frac{1}{\mu_2}(\cdots - \alpha_1 I)r_0, \\ \varphi_k\theta_{k-1} &= \frac{1}{\beta_{k+1}}[(\theta-\alpha_k)\varphi_{k-1}\theta_{k-1} - \gamma_k\varphi_{k-2}\theta_{k-1}], \quad k > 1 \end{aligned}$$

$$
\begin{aligned}
\text{or} \quad s_k^{BCS} &= \frac{1}{\beta_{k+1}}[(A-\alpha_k I)r_{k-1}^{BCS} - \gamma_k u_{k-1}^{BCS}], \quad k>1. \\
\varphi_0\theta_0 &= 1, \quad \text{or} \quad r_0^{BCS} = r_0, \\
\varphi_k\theta_k &= (1-\omega_k\theta)\varphi_k\theta_{k-1}, \quad \text{or} \quad r_k^{BCS} = (I-\omega_k A)s_k^{BCS}, \quad k \geq 1.
\end{aligned}
$$

As in CGS, α_k, β_k and γ_k are scalars from the unsymmetric Lanczos-BiCG algorithm and may be computed from the BiCGSTAB vectors by expanding the polynomials involved and using the simple fact

$$\tilde{r}_i^T \psi_j(A) r_0 = 0$$

for any polynomial ψ_j of degree $j < i$.

$$
\begin{aligned}
\text{Define} \quad \bar{\beta}_i &\equiv \prod_{j=1}^{i} \beta_{j+1}, \quad \bar{\omega}_i \equiv \prod_{j=1}^{i} \omega_j. \\
\rho_i &= \tilde{r}_{i-1}^T r_{i-1} = \tilde{r}_{i-1}^T (\frac{1}{\bar{\beta}_{i-1}} A^{i-1} r_0 + \cdots) \\
&= \frac{1}{\bar{\beta}_{i-1}} \tilde{r}_{i-1}^T A^{i-1} r_0, \quad i \geq 1. \\
\rho_i^{BCS} &= (\tilde{r}^{BCS})^T r_{i-1}^{BCS} \\
&= (\tilde{r}^{BCS})^T \varphi_{i-1}(A)\theta_{i-1}(A) r_0 \\
&= \tilde{r}_{i-1}^T (\bar{\omega}_{i-1} A^{i-1} r_0 + \cdots) \\
&= \bar{\omega}_{i-1} \tilde{r}_{i-1}^T A^{i-1} r_0. \\
\Rightarrow \rho_i &= \rho_i^{BCS} / (\bar{\omega}_{i-1}\bar{\beta}_{i-1}). \\
\gamma_1 &= 0 \\
\gamma_i &= \beta_i \rho_i / \rho_{i-1} = \rho_i^{BCS} / (\rho_{i-1}^{BCS} \omega_{i-1}), \quad i > 1. \\
\text{Now} \quad & (\tilde{r}^{BCS})^T (A r_{i-1}^{BCS} - \gamma_i u_{i-1}^{BCS}) \\
&= (\tilde{r}^{BCS})^T (A\varphi_{i-1}(A)\theta_{i-1}(A) - \gamma_i \varphi_{i-2}(A)\theta_{i-1}(A)) r_0 \\
&= (\tilde{r}^{BCS})^T (\beta_{i+1}\varphi_i(A) + \alpha_i(A)\varphi_{i-1}(A))\theta_{i-1}(A) r_0 \\
&= \beta_{i+1} \tilde{r}_i^T \theta_{i-1}(A) r_0 + \alpha_i \tilde{r}_{i-1}^T \theta_{i-1}(A) r_0 \\
&= 0 + \alpha_i \tilde{r}_{i-1}^T (\bar{\omega}_{i-1} A^{i-1} r_0 + \cdots)
\end{aligned}
$$

$$
\begin{aligned}
&= \alpha_i \bar{\omega}_{i-1} \tilde{r}_{i-1}^T A^{i-1} r_0 \\
&= \alpha_i \bar{\omega}_{i-1} \bar{\beta}_{i-1} \rho_i \\
&= \alpha_i \rho_i^{BCS} \\
\Rightarrow \alpha_i &= (\tilde{r}^{BCS})^T (A r_{i-1}^{BCS} - \gamma_i u_{i-1}^{BCS}) / \rho_i^{BCS}. \qquad (120)
\end{aligned}
$$

Finally $\beta_2 = -\alpha_1$ and $\beta_{i+1} = -(\alpha_i + \gamma_i)$, for $i > 1$, are immediately available.

We can now combine the above expressions to give the full computation for the r_k^{BCS}, s_k^{BCS} and u_k^{BCS} vectors — we call this the "BiCGSTAB process" — without yet showing the computation for the solution, which is given later.

New BiCGSTAB method

Problem: Solve $Ax = b$ for x when A is general nonsingular.

BiCGSTAB Process: We drop the superscript BCS from s_j and u_j, and write $\tilde{v} \equiv \tilde{r}^{BCS} = r_0; v_{j+1} \equiv r_j^{BCS}, \quad j \geq 0$, and for more efficient computation define $q_1 \equiv Av_1; \quad q_j \equiv Av_j - \gamma_j u_{j-1}$ for $j > 1$ and $t_j \equiv As_j$ for $j \geq 1$. Again, $1/\infty = 0$.

$$
\begin{aligned}
\rho_0 &\equiv \infty, \quad \omega_0 \equiv \infty, \quad \gamma_1 \equiv 0, \quad u_0 \equiv 0 \\
v_1 &= \tilde{v} = b \qquad (121) \\
\text{for} \quad j &= 1, 2, \ldots \\
&\rho_j = \tilde{v}^T v_j \\
&\gamma_j = -\rho_j / (\rho_{j-1} \omega_{j-1}) \\
&q_j = Av_j - \gamma_j u_{j-1} \qquad (122) \\
&\alpha_j = \tilde{v}^T q_j / \rho_j, \\
&\beta_{j+1} = -(\alpha_j + \gamma_j) \qquad (123) \\
&\beta_{j+1} s_j = q_j - \alpha_j v_j \qquad (124) \\
&t_j = As_j \\
&\omega_j = t_j^T s_j / t_j^T t_j
\end{aligned}
$$

$$u_j = v_j - \omega_j A v_j \tag{125}$$

$$v_{j+1} = s_j - \omega_j t_j. \tag{126}$$

Notice that, as required in (119),

$$\beta_{j+1}\tilde{v}^T s_j = \tilde{v}^T q_j - \alpha_j \tilde{v}^T v_j = \alpha_j \rho_j - \alpha_j \rho_j = 0, \quad j \geq 1,$$

follows immediately from the jth step.

We are again obliged to warn the reader that what follows is "work in progress", and the matrix representation is somewhat complicated, and probably not optimal.

Matrix Representation

If we use (122) and (125) substituted into (124) for the odd columns and (126) for the even columns the BiCGSTAB process becomes, in matrix form

$$\begin{aligned} \tilde{V}_{2k} &\equiv \begin{bmatrix} v_1 & s_1 & \cdots & v_k & s_k \end{bmatrix} \\ A\tilde{V}_{2k}G_{2k} &= \tilde{V}_{2k}F_{2k} - \frac{1}{\omega_k} v_{k+1} e_{2k}^T = \tilde{V}_{2k+1}F_{2k+1,2k} \end{aligned} \tag{127}$$

$$\text{where} \quad G_2 \equiv \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}, \quad G_{2k} \equiv \left[\begin{array}{cc|c} & & \omega_{k-1}\gamma_k \\ & G_{2k-2} & 0 \\ \hline & & 1 \\ & & & 1 \end{array}\right], \quad k > 1$$

$$F_2 \equiv \begin{bmatrix} \alpha_1 & \\ \beta_2 & \frac{1}{\omega_1} \end{bmatrix},$$

$$F_{2k} \equiv \left[\begin{array}{cc|c} & & \gamma_k \\ & F_{2k-2} & 0 \\ \hline & -\frac{1}{\omega_{k-1}} & \alpha_k \\ & & \beta_{k+1} \quad \frac{1}{\omega_k} \end{array}\right], \quad k > 1 \tag{128}$$

$$F_{2k+1,2k} \equiv \begin{bmatrix} F_{2k} \\ -\frac{1}{\omega_k} c_{2k}^T \end{bmatrix}.$$

Using (127) and solving for $F_{2k}\bar{y}_{2k} = e_1$, we consider iterates x_k as in (107),

$$\begin{aligned} x_k &= \bar{x}_{2k} = \bar{V}_{2k} G_{2k} \bar{y}_{2k} \\ \Rightarrow r_k &= b - Ax_k = b - A\bar{x}_{2k} = b - A\bar{V}_{2k} G_{2k} y_{2k} \\ &= b - (V_{2k} F_{2k} - \frac{1}{\omega_k} v_{k+1} c_{2k}^T)\bar{y}_{2k} \\ &= \bar{V}_{2k}(e_1 - F_{2k}\bar{y}_{2k}) + \frac{1}{\omega_k} v_{k+1} c_{2k}^T \bar{y}_{2k} \\ &= \frac{\bar{y}_{2k,2k}}{\omega_k} v_{k+1}. \end{aligned} \tag{129}$$

Subproblem: Solve $F_{2k}\bar{y}_{2k} = e_1$, where F_{2k} is upper Hessenberg as in (128). This does not have a solution if F_{2k} is singular, and at that time this implementation of the BiCGSTAB method would fail. However, we continue assuming that this is not the case.

Implementation: This may be solved using an LDU decomposition F_{2k}. The one which suits our purposes is shown below.

$$\begin{aligned} F_2 &= L_2 \bar{D}_2 \bar{U}_2 \\ &= \begin{bmatrix} 1 & \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -\beta_2 & \\ & \frac{1}{\omega_1} \end{bmatrix} \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} = \begin{bmatrix} -\beta_2 & \\ \beta_2 & \frac{1}{\omega_1} \end{bmatrix}. \end{aligned} \tag{130}$$

$$F_{2k} = L_{2k} \bar{D}_{2k} \bar{U}_{2k}$$

$$L_{2k} = \left[\begin{array}{cc|c} & L_{2k-2} & \\ \hline & -1 & 1 \\ & & -1 \quad 1 \end{array}\right], \quad \bar{D}_{2k} = \left[\begin{array}{cc|c} & \bar{D}_{2k-2} & \\ \hline & & -\beta_{k+1} \\ & & \frac{1}{\omega_k} \end{array}\right],$$

$$\bar{U}_{2k} = \left[\begin{array}{cc|c} & & -\frac{\gamma_k}{\beta_k} \\ & \bar{U}_{2k-2} & \omega_{k-1}\gamma_k \\ \hline & & 1 \\ & & & 1 \end{array}\right]$$

$$\Rightarrow F_{2k} = \left[\begin{array}{cc|c} & & \gamma_k \\ & L_{2k-2}\bar{D}_{2k-2}\bar{U}_{2k-2} & 0 \\ \hline & -\frac{1}{\omega_{k-1}} & -(\gamma_k+\beta_{k+1}) \\ & & \beta_{k+1} \quad \frac{1}{\omega_k} \end{array}\right].$$

Our normalizers β_{j+1} are now verified by equating the right sides of the above equations with F_2 and F_{2k}. So

$$\begin{aligned} \alpha_1 &= -\beta_2; \quad \alpha_k = -(\gamma_k + \beta_{k+1}), \quad k > 1 \\ \Rightarrow \beta_2 &\equiv -\alpha_1; \quad \beta_{k+1} \equiv -(\alpha_k + \gamma_k), \quad k > 1. \end{aligned} \tag{131}$$

This was the choice of β_{k+1} that we made in § 4.6. As long as it is nonzero, it is a satisfactory normalizer for the process, and $\bar{U}_{2k+2}$ exists. For ease of calculation, we make the following definitions.

$$\bar{W}_{2k} \equiv \left[\begin{array}{cccc} \bar{w}_1 & \bar{w}_2 & \cdots & \bar{w}_{2k-1} \quad \bar{w}_{2k} \end{array}\right] \equiv \bar{V}_{2k} G_{2k} \bar{U}_{2k}^{-1}$$

$$\text{and} \quad \bar{z}_{2k} \equiv \left[\begin{array}{c} \bar{\zeta}_1 \\ \bar{\zeta}_2 \\ \vdots \\ \bar{\zeta}_{2k} \end{array}\right] \equiv \bar{U}_{2k}\bar{y}_{2k}$$

$$\Rightarrow \bar{\zeta}_{2k} = \bar{y}_{2k,2k}. \tag{132}$$

So our system is transformed to

$$L_{2k}\bar{D}_{2k}\bar{z}_{2k} = e_1, \quad \bar{x}_k = \bar{W}_{2k}\bar{z}_{2k}.$$

Solving

$$L_{2k}\bar{D}_{2k}\bar{z}_{2k} = \begin{bmatrix} -\beta_2\bar{\zeta}_1 \\ \beta_2\bar{\zeta}_1 + \frac{1}{\omega_1}\bar{\zeta}_2 \\ -\frac{1}{\omega_1}\bar{\zeta}_2 - \beta_3\bar{\zeta}_3 \\ \vdots \\ -\frac{1}{\omega_{k-1}}\bar{\zeta}_{2k-2} - \beta_{k+1}\bar{\zeta}_{2k-1} \\ \beta_{k+1}\bar{\zeta}_{2k-1} + \frac{1}{\omega_k}\bar{\zeta}_{2k} \end{bmatrix} = e_1$$

$$\begin{aligned} \Rightarrow \bar{\zeta}_1 &= -1/\beta_2, \quad \bar{\zeta}_2 = \omega_1 \\ \bar{\zeta}_{2k-1} &= -\frac{1}{\beta_{k+1}\omega_{k-1}}\bar{\zeta}_{2k-2}, \quad k > 1 \\ \bar{\zeta}_{2k} &= -\beta_{k+1}\omega_k\bar{\zeta}_{2k-1} = \frac{\omega_k}{\omega_{k-1}}\bar{\zeta}_{2k-2} = \omega_k, \quad k > 1 \\ \bar{\zeta}_{2k-1} &= -\frac{1}{\beta_{k+1}}, \quad k > 1. \end{aligned}$$

Consider the odd and even columns of $\bar{W}_{2k}\bar{U}_{2k} = \bar{V}_{2k}G_{2k}$.

$$\bar{w}_1 = v_1 \tag{133}$$

$$\bar{w}_{2k-1} = \omega_{k-1}\gamma_k(v_{k-1} - s_{k-1}) + v_k + \frac{\gamma_k}{\beta_k}\bar{w}_{2k-3}, \quad k > 1 \tag{134}$$

$$\bar{w}_{2k} = s_k, \quad k \geq 1.$$

This causes us to redefine $\bar{W}_{2k} \equiv \begin{bmatrix} p_1 & s_1 & \cdots & p_k & s_k \end{bmatrix}$ and rewrite (133) and (134) as

$$\begin{aligned} p_1 &= v_1 \\ p_k &= v_k + \omega_{k-1}\gamma_k(v_{k-1} - s_{k-1}) + \frac{\gamma_k}{\beta_k}p_{k-1}, \quad k > 1. \end{aligned}$$

$$\text{Now} \quad r_k = \frac{\bar{y}_{2k,2k}}{\omega_k} = \frac{\bar{\zeta}_{2k}}{\omega_k}v_{k+1} = v_{k+1} \tag{135}$$

$$\begin{aligned} \text{and} \quad x_k &= \bar{W}_{2k-1}\bar{z}_{2k-2} + \bar{\zeta}_{2k-1}\bar{w}_{2k-1} + \bar{\zeta}_{2k}\bar{w}_{2k} \\ &= x_{k-1} - \frac{1}{\beta_{k+1}}p_k + \omega_k s_k \end{aligned} \tag{136}$$

so this BiCGSTAB process produces vectors v_{j+1} which are the residual vectors r_j of the method.

It remains to examine under what conditions β_{j+1} can be zero. Supose $\beta_2, \ldots, \beta_j$ obtained from (131) are nonzero. The with $e^T \equiv [1, \ldots, 1]$, we have

$$\begin{aligned} e^T F_{2j-1} &= [0, \ldots, \frac{1}{\omega_{j-1}} - \frac{1}{\omega_{j-1}}, \alpha_j + \gamma_j] \\ &= [0, \ldots, 0, \alpha_j + \gamma_j], \end{aligned}$$

which willl be nonzero if F_{2j-1} is nonsingular. However if $\alpha_j + \gamma_j = 0$ then F_{2j-1} is singular, and both this and the standard CGS method fails. Otherwise it is clear that the $L_{2k}\bar{D}_{2k}\bar{U}_{2k}$ 'factorization' is possible. However small β_k could cause large element growth in $\bar{U}_{2k}$.

Equivalence of this method with the vector form: The BiCGSTAB method — slightly altered from 'Templates' — is as follows.

$r_0 = b$

$\tilde{r} = r_0$

for $i = 1, 2, \ldots$

  $\rho_{i-1} = \tilde{r}^T r_{i-1}$

  if $\rho_{i-1} = 0$, method fails

  if $i = 1$

    $p_i = r_{i-1}$

  else

    $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$

    $p_i = r_{i-1} + \beta_{i-1}(p_{i-1} - \omega_{i-1} w_{i-1})$

  $w_i = A p_i$

  $\alpha_i = \rho_{i-1}/\tilde{r}^T w_i$

  $s_i = r_{i-1} - \alpha_i w_i$

  check norm of s; if small enough: set $x_i = x_{i-1} + \alpha_i p_i$ and stop

  $t_i = A s_i$

  $\omega_i = t_i^T s_i / t_i^T t_i$

$x_i = x_{i-1} + \alpha_i p_i + \omega_i s_i$

$r_i = s_i - \omega_i t_i$

check for convergence; continue if necessary

for continuation it is necessary that $\omega_i \neq 0$

end

To avoid confusion we will use the superscript ^{B} to distinguish 'Templates' terms from those here, where it will also make life easier if we re-index two of the 'Templates' terms as follows:

$$\rho_i^B \equiv \text{'Templates'}\rho_{i-1},$$
$$\beta_i^B \equiv \text{'Templates'}\beta_{i-1}.$$

We now write the algorithms side by side for comparison, where in this new BiCGSTAB algorithm we replace v_{i+1} by r_i, see (135). To handle the $i = 1$ case we define

BiCGSTAB	new BiCGSTAB
$\rho_0^B = 1, \alpha_0^B = 0, \omega_0^B = 0$	$\rho_0 = 1, \beta_1 = 1, \omega_0 = 0$
$p_0^B = 0$	$v_0 = 0, s_0 = 0, p_0 = 0, u_0 = 0$

Then for $i = 1, 2, 3, \ldots$ we have

BiCGSTAB	new BiCGSTAB
$\rho_i^B = \tilde{r}^{BT} r_{i-1}^B$	$\rho_i = \tilde{v}^T r_{i-1}$
if $i = 1$	if $i = 1$
$\beta_1^B = 0$	$\gamma_1 = 0$
else	else
$\beta_i^B = (\rho_i^B / \rho_{i-1}^B)(\alpha_{i-1}^B / \omega_{i-1}^B)$	$\gamma_i = -(\rho_i / \rho_{i-1})(1/\omega_{i-1})$
$p_i^B = r_{i-1}^B + \beta_i^B (p_{i-1}^B - \omega_{i-1}^B A p_{i-1}^B)$	$p_i = r_{i-1} + \frac{\gamma_i}{\beta_i} p_{i-1}$
	$+\omega_{i-1}\gamma_i (r_{i-2} - s_{i-1})$

$\alpha_i^B = \rho_i^B / \tilde{r}^{BT} A p_i^B$	$\alpha_i = \tilde{v}^T (A r_{i-1} - \gamma_i u_{i-1})/\rho_i$
	$\beta_{i+1} = -(\alpha_i + \gamma_i)$
$s_i^B = r_{i-1}^B - \alpha_i^B A p_i^B$	$s_i = (A r_{i-1} - \alpha_i r_{i-1} - \gamma_i u_{i-1})/\beta_{i+1}$
$\omega_i^B = (A s_i^B)^T s_i^B / (A s_i^B)^T (A s_i^B)$	$\omega_i = (A s_i)^T s_i / (A s_i)^T (A s_i)$
	$u_i = r_{i-1} - \omega_i A r_{i-1}$
$x_i^B = x_{i-1}^B + \alpha_i^B p_i^B + \omega_i^B s_i^B$	$x_i = x_{i-1} - 1/\beta_{i+1} p_i + \omega_i s_i$
$r_i^B = s_i^B - \omega_i^B A s_i^B$	$r_i = s_i - \omega_i A s_i$

There are noticeable differences between the two algorithms — the new BiCGSTAB algorithm has more complicated expressions for p_i and s_i and also forms an intermediate vector u_i, which the 'Templates' algorithm does not. However relationships exist between the two; the exact equivalences are

$$\begin{aligned} \gamma_i/\beta_i &= \beta_i^B, \quad i > 1 \\ \rho_i &= \rho_i^B, \quad -1/\beta_{i+1} = \alpha_i^B, \quad \omega_i = \omega_i^B \\ p_i &= p_i^B, \quad s_i = s_i^B, \quad u_i = r_{i-1}^B - \omega_i A r_{i-1}^B \\ x_i &= x_i^B, \quad r_i = r_i^B. \end{aligned} \tag{137}$$

A straightforward examination shows this is true for $i = 1$. Suppose it is true for $i = 1, 2, \ldots, k-1$. Then it is clear that

$$\rho_k = \rho_k^B, \quad \gamma_k/\beta_k = \beta_k^B.$$

The following will be useful in proving the rest.

$$P_k = \tilde{W}_{2k} I_{2k,k}$$

where $I_{2k,k} \equiv \left[\begin{array}{c|c} I_{2k-2,k-1} & \\ \hline & \begin{matrix} 1 \\ 0 \end{matrix} \end{array}\right]$

$$\text{So} \quad AP_k = A\bar{W}_{2k}I_{2k,k} = A\bar{V}_{2k}G_{2k}\bar{U}_{2k}^{-1}I_{2k,k}$$

$$= \bar{V}_{2k+1}F_{2k+1,2k}\bar{U}_{2k}^{-1}I_{2k,k} = \bar{V}_{2k+1}L_{2k+1,2k}\bar{D}_{2k}I_{2k,k} \quad (138)$$

$$= \bar{V}_{2k+1}\left[\begin{array}{c|c} J_{2k-1,k-1} & \\ \hline & \begin{array}{c} -\beta_{k+1} \\ \hline \beta_{k+1} \\ 0 \end{array} \end{array}\right]. \quad (139)$$

To show $p_k = p_k^B$, consider the i-th column of (139).

$$Ap_i = \beta_{i+1}(s_i - r_{i-1}), \quad i = 1, 2, \ldots, k \quad (140)$$

$$p_k = r_{k-1} + \omega_{k-1}\gamma_k(r_{k-2} - s_{k-1}) + \gamma_k/\beta_k p_{k-1}$$

$$= r_{k-1}^B - \omega_{k-1}^B\gamma_k/\beta_k Ap_{k-1} + \beta_k^B p_{k-1}^B$$

$$= r_{k-1}^B - \omega_{k-1}^B\beta_k^B Ap_{k-1}^B + \beta_k^B p_{k-1}^B = p_k^B.$$

To show $-1/\beta_{k+1} = \alpha_k^B$, consider the scalar product of $\tilde{v}$ with (140)

$$\tilde{v}^T Ap_i = \beta_{i+1}\tilde{v}^T(s_i - r_{i-1})$$

$$= -\rho_i\beta_{i+1}, \quad i = 1, 2, \ldots, k.$$

$$\text{So} \quad \alpha_k^B = \rho_k^B/\tilde{r}^T Ap_k^B = \rho_k/\tilde{v}^T Ap_k = -1/\beta_{k+1}.$$

(140) makes it easy to prove $s_k = s_k^B$.

$$s_k^B = r_{k-1}^B - \alpha_k^B Ap_k^B = r_{k-1} + 1/\beta_{k+1}Ap_k$$

$$= r_{k-1} + s_k - r_{k-1} = s_k.$$

Finally, it is obvious that

$$\omega_k = \omega_k^B, \quad x_k = x_k^B, \quad r_k = r_k^B.$$

The induction proof (137) is complete, showing that in theory these algorithms not only compute the same iterates, but also several of the intermediate quantities are identical. However the algorithms are clearly not

identical, as the use of different vectors shows, and may have different numerical behaviours. This suggests a matrix development of BiCGSTAB would be useful, but we leave this for future work.

Reference: [vdV92]

5 Conclusion

From the point of view of developing the research, the most interesting and satisfactory applications of the formalism stated in § 1.1 were to the methods of Conjugate Gradients and BiConjugate Gradients. In these, connections with the underlying symmetric and unsymmetric Lanczos processes respectively were exposed. In each case two methods were developed — one with unit norm vectors spanning $\mathcal{K}^k(A;b)$ and one which produced the residual vectors of the method to span the Krylov subspace. In the case of BiCG, the Krylov process which produced the residual vectors was rather surprisingly shown to use the same coefficients in the formation of the "shadow residuals", as opposed to the familiar unsymmetric Lanczos process which "transposed" the coefficients. It was the second method in each case which was shown to be mathematically equivalent to the "vector" form, since the resulting algorithm more closely resembled the vector form. An understanding of the quantities used in the vector form of the algorithm was obtained and in general the algorithm was clarified.

The methods of SYMMLQ, MINRES, LSQR, and GMRES were originally implicitly developed using the formalism. Thus the Krylov processes and associated subproblems were easy to determine for these and there was no difference between the algorithm derived from the matrix development and the corresponding vector form.

The method of Quasi-Minimal Residuals as developed in [FN91] also implicitly used the formalism. The Krylov process and subproblem were obvious and implementation details were also given in the paper. However, the 'Templates' version of QMR differs from that in the original paper — although theoretically the two are the same, the vector form has been derived from a subproblem whose solution implementaion is less strightforward, in

the interests of time and storage of the algorithm. As future work, a matrix development of that 'Templates' vector form of QMR to determine the Krylov process and subproblem and hence show the resulting method is QMR would make § 4.7 more complete.

Attempts to apply the formalism to CGS and BiCGSTAB, algorithms which double the complexity of BiCG in an attempt to halve the time, required a small modification to the approach suggested in § 4. Since these two methods produce residual vectors which are elements of every other Krylov subspace, it was necessary to consider auxiliary vectors, elements of the missed subspaces, as well. Using both sets of vectors, we achieved a matrix equation of the form

$$\begin{aligned} A\bar{V}_K G_K &= \bar{V}_{K+1} F_{K+1,K} \\ \Rightarrow A\bar{V}_K &= \bar{V}_{K+1} H_{K+1,K} \qquad (141) \\ \text{where} \quad H_{K+1,K} &\equiv F_{K+1,K} G_K^{-1} \quad \text{is upper Hessenberg} \\ \text{and} \quad K &\equiv 2k. \end{aligned}$$

The form of (141) is exactly the same as for the general Krylov process (3) in § 2.

The next difference came in the iterates we used. The original idea was to form iterates $x_k = V_k y_k$, but here we chose

$$\begin{aligned} x_k &= \bar{x}_K = \bar{V}_K G_K \bar{y}_K \qquad (142) \\ &= \bar{V}_K \bar{y}_K \qquad (143) \\ \text{where} \quad \bar{y}_K &= G_K \bar{y}_K, \end{aligned}$$

thus a simple matter of relabelling (142) gives (143) just as (47) in § 4. With this slightly different approach, attempts to reveal the process and subproblem were successful although not as elegant as might have been

hoped. As was stated in those two subsections, the work done is really "work in progress" and certainly further improvements may be made.

On the whole the formalism appears to be very useful in the analysis of algorithms and definitely makes the more mysterious "vector" forms much easier to understand and learn. Unfortunately, as was observed in the cases of CGS and BiCGSTAB, it appears somewhat difficult in the more complex algorithms to determine an optimally clear, simple and elegant division into process and problem.

References

[Arn51] W.E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[BBC+94] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM Publications, Philadelphia, Pennsylvania 19104-2688, 1994. There may be a later version.

[FN91] R.W. Freund and N.M. Nachtigal. QMR: A quasi-minimal residual method for non-hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[GK65] G.H. Golub and W. Kahan. Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal.*, 2:205–224, 1965.

[GL89] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, second edition.* The Johns Hopkins University Press, Baltimore, Maryland 21211, 1989.

[Hou74] A.S. Householder. *The Theory of Matrices in Numerical Analysis.* Dover Publications, New York, 1974. Originally published by Blaisdell, NY, 1964.

[HS52] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[Kry31] A.N. Krylov. O čislennom rešenii uravnenija, kotorym v techničeskih voprasah opredeljajutsja častoty malyh kolebanii material'nyh sistem. *Izv. Akad. Nauk SSSR Otd. Mat. Estest.*, pages 491–539, 1931.

[Lan50] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Stand.*, 45:255–282, 1950.

[Lan52] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Stand.*, 49:33–53, 1952.

[Pai71] C.C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices.* PhD thesis, London University, London, England, 1971.

[Pai72] C.C. Paige. Computational variants of the Lanczos method for the eigenproblem. *J. Inst. Math. Appl.*, 10:373–381, 1972.

[Pai94] C.C. Paige. Krylov subspace processes, Krylov subspace methods, and iteration polynomials. In M. Chu, R. Plemmons, D. Brown, and D. Ellison, editors, *Proceedings of the Lanczos Centenary Conference, Raleigh, NC, Dec. 1993*, pages 83–92, Philadelphia, Pa, 1994. SIAM Publications.

[PPdV95] C.C. Paige, B.N. Parlett, and H.A. Van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numerical Linear Algebra with Applications*, 1995. To appear, May 1995.

[PS75] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[PS82] C.C. Paige and M.A. Saunders. LSQR: An algorithm for sparse linear eqautions and sparse least squares. *ACM Transactions on Mathematical Software*, 8:43–71, 1982.

[Rei71] J.K. Reid. On the method of conjugate gradients for the solution of large sparse linear equations. In J.K. Reid, editor, *Large Sparse Sets of Linear Equations*, pages 231–254. Academic Press, New York, 1971.

[Son89] P. Sonneveld. CGS: A fast lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.

[SS86] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[vdV92] H.A. 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.