

1 Introduction

Iterative methods for solving linear systems $Ax = b$ become necessary when $A \in \mathbb{R}^{n \times n}$ is too large to factorize directly. The meaning of *too large* depends on the context, but $n = 10000$ up to $n = 10^8$ is typical. In optimization, the following symmetric (but possibly indefinite) examples arise in computing search directions:

- Newton's method for unconstrained optimization: $H\Delta x = -g$.
- Newton's method for optimization with linear constraints $Jx = b$:
Solve $Z^T H Z v = -Z^T g$ and set $\Delta x = Zv$, where Z spans the null space of the constraint matrix ($JZ = 0$).
- KKT systems with linearized constraints:
$$\begin{pmatrix} -H & J^T \\ J & \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} g - J^T y \\ -c - Jx \end{pmatrix}.$$

The *conjugate-gradient method* (CG) is the prototype solver for $Ax = b$ when A is symmetric and positive definite. Distinguishing features of CG follow:

- A is regarded as an *operator*. For various vectors v , CG asks for the matrix-vector product $y = Av$. This is the only way that A is defined.
- Only a few work n -vectors of storage are needed to generate each approximate solution x_k ($k = 1, 2, \dots$).
- With exact arithmetic, CG would terminate in at most n iterations. In practice it may need *far fewer* iterations if A has clustered eigenvalues, or *far more* if we are not so lucky. Of course the first situation is preferable.
- Favorable eigenvalue distributions can be achieved by finding a *preconditioner* M such that $M = CC^T \approx A$ (in some sense) and solving a transformed system $\bar{A}\bar{x} = \bar{b}$, where \bar{A} is the operator $C^{-1}AC^{-T} \approx I$ and the remaining quantities are obtained by solving $C\bar{b} = b$ and $C^T x = \bar{x}$.
- The matrix-vector product $y = \bar{A}v$ means "Solve $C^T w_1 = v$, form $w_2 = Aw_1$, and solve $Cy = w_2$ ". Thus it must be possible to solve with C and C^T reasonably efficiently (as well as multiplying by A). The simplest example is *diagonal preconditioning* with $C = \text{diag}(\sqrt{A_{jj}})$.
- PCG (preconditioned CG) is a rearrangement of CG that allows solves with M itself, rather than C and C^T separately. Diagonal preconditioning then means working with the preconditioner $M = \text{diag}(A)$ in MATLAB notation.

2 Lanczos-based Methods for Symmetric Systems

We review three methods for solving symmetric systems $Ax = b$. As described in [6], the methods CG, MINRES, and SYMMLQ are based on the Lanczos process [3] for tridiagonalizing A . A helpful framework for viewing such methods has been suggested by Paige [5]:

An iterative *process* generates certain quantities from the data. At each iteration a *subproblem* is defined, suggesting how those quantities may be combined to give a new estimate of the required solution. Different subproblems define different *methods* for solving the original problem. Different ways of solving a subproblem lead to different *implementations* of the associated method.

Typically the subproblems may be solved efficiently and stably (though stability questions are sometimes overlooked). The numerically difficult aspects are usually introduced by the *process*.

2.1 The Lanczos Process

$\text{Tridiag}(A, b) \rightarrow (T_k, V_k)$ denotes the following process. Given a symmetric matrix A and a starting vector b , the Lanczos process generates vectors v_k and scalars α_k, β_k ($k = 1, 2, \dots$) according to these steps:

1. Set $\beta_1 v_1 = b$. (This means $\beta_1 \leftarrow \|b\|_2$ and then $v_1 \leftarrow b/\beta_1$, but exit if $\beta_1 = 0$.)
2. For $k = 1, 2, \dots$, set

$$\begin{aligned} w &= Av_k, \\ \alpha_k &= v_k^T w, \\ \beta_{k+1} v_{k+1} &= w - \alpha_k v_k - \beta_k v_{k-1}. \end{aligned}$$

After k steps with $\beta_1, \dots, \beta_k > 0$, the situation may be summarized as

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T = V_{k+1} H_k, \quad (1)$$

where e_k is the k th unit vector, $V_k = (v_1 \ v_2 \ \dots \ v_k)$, T_k is tridiagonal, and H_k is also tridiagonal with one extra row:

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & & \beta_k & \alpha_k \end{pmatrix}, \quad H_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & & \beta_k & \alpha_k \\ & & & & \beta_{k+1} \end{pmatrix} = \begin{pmatrix} T_k \\ \beta_{k+1} e_k^T \end{pmatrix}.$$

With exact arithmetic the columns of V_k would be orthonormal for each k until $\beta_{k+1} = 0$. In practice, $V_k^T V_k = I$ quickly fails to hold (unless v_{k+1} is reorthogonalized with respect to previous vectors), but relation (1) is *accurate to machine precision*.

From the way the Lanczos vectors $\{v_1, v_2, \dots\}$ are generated, it is clear that v_k for each k lies in the *Krylov subspace* $\mathcal{K}(A, b, k) \equiv \mathcal{K}_k = \text{span}\{b, Ab, \dots, A^{k-1}b\}$.

2.2 Lanczos with Shifts

Let δ be a given scalar. The following property is readily proved from (1). It shows that T_k and V_k are (almost) independent of a shift δ in the diagonals of A .

Result 1 *If $\text{Tridiag}(A, b) \rightarrow (T_k, V_k)$, then $\text{Tridiag}(A + \delta I, b) \rightarrow (T_k + \delta I, V_k)$.*

2.3 CG, MINRES, and SYMMLQ

Table 1 lists three ways to choose “optimal” points within each Krylov subspace \mathcal{K}_k (i.e., points of the form $x_k = V_k y_k$ for some vector y_k). The three choices lead to three *methods* for solving $Ax = b$, namely CG, MINRES, and SYMMLQ. Note that the CG method is not meaningful if the quadratic form is unbounded below. This means that A must be positive-definite for CG.

From (1) we see that the *residual vector* associated with a point $x_k \in \mathcal{K}_k$ is

$$r_k \equiv b - Ax_k \quad (2)$$

$$= \beta_1 v_1 - AV_k y_k$$

$$= V_{k+1}(\beta_1 e_1 - H_k y_k)$$

$$= V_{k+1} t_{k+1}, \quad (3)$$

$$\text{where } t_{k+1} \equiv \beta_1 e_1 - H_k y_k. \quad (4)$$

This suggests that $\|r_k\|$ will be small if y_k makes t_{k+1} small by some measure. Indeed, we find that Table 1’s subproblems for x_k lead to the corresponding subproblems for y_k shown in Table 2 (which also shows the factorizations needed to solve the subproblems). The CG subproblem makes $t_{k+1} = 0$ everywhere except its last element, while the MINRES subproblem is more balanced in minimizing $\|t_{k+1}\|$. The SYMMLQ subproblem makes $t_{k+1} = 0$ everywhere except its last *two* elements.

If A is positive definite, so is T_k for all k . CG can therefore obtain Cholesky factors of each T_k . MINRES uses the QR factorization of H_k , and is applicable to any symmetric A (including singular systems if suitable stopping criteria are implemented). SYMMLQ uses the same QR factorization, disguised as the LQ factorization of H_k^T , and is again applicable to any symmetric A , except that $Ax = b$ must be compatible. MINRES-QLP works with a two-sided orthogonal factorization of H_k for greater reliability on ill-conditioned or singular systems (see Sou-Cheng Choi’s thesis [1]). The QLP name comes from Stewart [11].

Note that all elements of y_k may change in y_{k+1} . Also, we don’t wish to store all of V_{k+1} in order to form $V_{k+1} y_{k+1}$. Thus, each method computes certain quantities W_k and z_k that allow the solution estimates to be *updated*. Table 3 shows all the possibilities we can think of. (Tables 2–3 are from [9], except for the recent MINRES-QLP entries.)

Table 1: Minimization properties for three methods for solving $Ax = b$. They seek points x_k that give small residual vectors $r_k \equiv b - Ax_k$.

Method	Definition of optimal x_k in Krylov subspace
CG	$\min \frac{1}{2}x_k^T A x_k - b^T x_k$ s.t. $x_k \in \mathcal{K}_k$ $\equiv \min \ r_k\ _{A^{-1}}^2$ s.t. $x_k \in \mathcal{K}_k$
MINRES	$\min \ r_k\ ^2$ s.t. $x_k \in \mathcal{K}_k$
SYMMLQ	$\min \ x_k\ ^2$ s.t. $x_k \in \mathcal{K}_k, r_k \perp \mathcal{K}_{k-1}$ $\equiv \min \ x - x_k\ ^2$ s.t. $x_k \in A\mathcal{K}_k$ (?)

Table 2: Subproblems defining y_k and $x_k = V_k y_k$ for four methods.

Method	Subproblem	Factorization	Estimate of x
CG	$T_k y_k = \beta_1 e_1$	$T_k = L_k D_k L_k^T$	$x_k^C = V_k y_k$
MINRES	$\min \ H_k y_k - \beta_1 e_1\ $	$Q_k H_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix}$	$x_k^M = V_k y_k$
SYMMLQ	$\min \ y_k\ $ s.t. $H_{k-1}^T y_k = \beta_1 e_1$	$H_{k-1}^T Q_{k-1}^T = (L_{k-1} \ 0)$	$x_k^L = V_k y_k$
MINRES-QLP	$\min \ y_k\ $ s.t. $\ H_k y_k - \beta_1 e_1\ = \min$	$R_k P_k = \bar{L}_k$	$x_k^Q = V_k y_k$

Table 3: Definition of W_k and z_k such that $x_k = V_k y_k = W_k z_k$.

	W_k	z_k	Estimate of x
CG	$V_k L_k^{-T}$	$L_k D_k z_k = \beta_1 e_1$	$x_k^C = W_k z_k$
MINRES	$V_k R_k^{-1}$	$\begin{pmatrix} z_k \\ \bar{\zeta}_{k+1} \end{pmatrix} = Q_k \begin{pmatrix} \beta_1 e_1 \\ 0 \end{pmatrix}$	$x_k^M = W_k z_k$
SYMMLQ	$\bar{W}_k = V_k Q_{k-1}^T$ $= (W_{k-1} \ \bar{w}_k)$	$L_{k-1} z_{k-1} = \beta_1 e_1$ $\bar{\zeta}_k = 0$	$x_k^L = W_k \bar{z}_k$ $= W_{k-1} z_{k-1}$
MINRES-QLP	$V_k P_k$	$\bar{L}_k \bar{z}_k = z_k$	$x_k^Q = W_k \bar{z}_k$

Note that $V_k(\beta_1 e_1) = b$ exactly for all k because v_1 is a multiple of b . Thus, the relations $r_k = V_{k+1}t_{k+1}$ and $t_{k+1} = \beta_1 e_1 - H_k y_k$ (3)–(4) hold accurately for any y_k even though the columns of V_{k+1} lose orthogonality. Since

$$\|r_k\| \leq \|V_{k+1}\| \|\beta_1 e_1 - H_k y_k\|$$

with $\|V_{k+1}\| = O(1)$ and $\|\beta_1 e_1 - H_k y_k\|$ tending to decrease for the given choices of y_k , we can expect $\|r_k\|$ to become small eventually.

The CG iteration CG treats $T_k y_k = \beta_1 e_1$ as $L_k D_k (L_k^T y_k) = \beta_1 e_1$, defines $z_k \equiv L_k^T y_k$, and solves the lower-triangular systems

$$\begin{aligned} L_k D_k z_k &= \beta_1 e_1, & z_k &= \begin{pmatrix} z_{k-1} \\ \zeta_k \end{pmatrix} \\ L_k W_k^T &= V_k^T, & W_k^T &= \begin{pmatrix} W_{k-1}^T \\ w_k^T \end{pmatrix} \end{aligned}$$

by computing *only the last elements of the solutions* at each iteration k , taking advantage of the fact that the preceding parts of z_k and W_k have already been computed. For example, since L_k is *lower bidiagonal with unit diagonals*, we can form $w_k = v_k - \lambda_k w_{k-1}$ efficiently, where λ_k is the $(k, k-1)$ element of L_k . Thus,

$$\begin{aligned} x_k &= V_k y_k \\ &= W_k L_k^T y_k \\ &= W_k z_k \\ &= W_{k-1} z_{k-1} + w_k \zeta_k \\ &= x_{k-1} + \zeta_k w_k \end{aligned}$$

can also be formed cheaply. The vectors in V_{k-2} and W_{k-1} are no longer needed.

Although we don't want all of y_k , we see from $L_k^T y_k = z_k$ that the last element of y_k is $\eta_k = \zeta_k$. Also from (3)–(4) and the fact that CG makes t_{k+1} zero except for its last element $\tau_{k+1} \equiv e_{k+1}^T t_{k+1}$, we see that the CG residual vector satisfies $r_k^L = \tau_{k+1} v_{k+1}$ and hence $\|r_k^L\| = |\tau_{k+1}| = |-\beta_{k+1} \eta_k| = |\beta_{k+1} \zeta_k|$. Thus when CG is implemented this way, we have an accurate estimate of $\|r_k^C\|$ at essentially no cost.

The MINRES iteration By definition, the MINRES point x_k^M solves the problem

$$\min_y \|r_k\| \quad \text{such that} \quad x_k = V_k y,$$

so that $\|r_k^M\|$ decreases monotonically, and there is no difficulty if the system is incompatible. Many users prefer MINRES for these reasons. The iteration is similar to CG in solving $R_k^T W_k^T = V_k^T$ for each w_k in turn and updating $x_k = x_{k-1} + \zeta_k w_k$. There is more work and storage because R_k^T is lower *tridiagonal*. A numerical concern is that the columns of W_k could be large if some of the R_k are ill-conditioned.

The SYMMLQ iteration In contrast, SYMMLQ's point x_k^L solves

$$\min_y \|x_k\| \quad \text{such that} \quad x_k = V_k y \quad \text{and} \quad V_k^T r_k = 0,$$

so that $\|x_k^L\|$ increases and the system must be compatible. It also solves

$$\min_t \|x - x_k\| \quad \text{such that} \quad x_k = AV_k t$$

[2, 4], so that the error norm $\|x - x_k^L\|$ decreases monotonically.

The SYMMLQ solutions $x_k^L = W_{k-1} z_{k-1} = x_{k-1}^L + \zeta_{k-1} w_{k-1}$ are accumulated as a sequence of theoretically *orthogonal* steps. Although the columns of V_k and W_k are not orthonormal in practice, we will always have $\|w_k\| \approx 1$, so that forming x_k^L should involve less cancellation error than in MINRES (where the columns of $W_k = V_k R_k^{-1}$ could be large).

By observation, $\|r_k\|$ is often much larger for SYMMLQ than for the other methods. This is not cause for concern. It's a sign that SYMMLQ is stepping around points that would be troublesome for CG. Since the residual norms can be estimated cheaply, SYMMLQ has provision for *transferring* to the CG point upon termination if the residual is then smaller. Thus, if $\|r_{k+1}^C\| < \|r_k^L\|$, SYMMLQ takes a final step of the form $x_k^C = x_k^L + \bar{\zeta}_k \bar{w}_k$, where the last two items are already known.

Note that after k iterations, SYMMLQ has solved just a single triangular system $L_{k-1} z_{k-1} = \beta_1 e_1$, and this is the only place where ill-conditioning in A becomes evident. We therefore believe that SYMMLQ is the method of choice for indefinite compatible systems. MINRES-QLP should be comparable in reliability at the cost of slightly more work and storage per iteration, and it handles singular systems well.

MINRES-QLP The effects of rounding errors on the convergence of CG, MINRES, and SYMMLQ have been analyzed by Sleijpen et al. [10]. Some numerical examples confirm that MINRES may not achieve small $\|r_k\|$ on compatible systems when A is very ill-conditioned.

MINRES-QLP is the only method that returns the minimum-length solution on singular incompatible systems $Ax \approx b$. It is significantly more complex (see Choi [1]) but can be more reliable than MINRES when A is ill-conditioned. In [1] it is anticipated that the rounding errors in MINRES's solution of the n independent ill-conditioned triangular systems $R_k^T W_k^T = V_k^T$ (i.e., in the n rows of W_k) are more significant than in MINRES-QLP's solution of the *single* ill-conditioned system $\bar{L}_k \bar{z}_k = z_k$, as in SYMMLQ.

2.4 Estimation of Norms

At iteration k of the above solvers, estimates of $\|r_k\|$, $\|Ar_k\|$, $\|x_k\|$, $\|A\|$, and $\text{cond}(A)$ are needed in order to implement reliable stopping rules. The estimates have been studied most fully for MINRES and MINRES-QLP in Choi [1]. In particular, $\|A\|_2 \approx \|T_k\|_2$ or $\|H_k^T H_k\|_1^{1/2}$ are reasonable estimates that can be estimated cheaply as the iterations proceed. Different solvers estimate the other quantities in various ways.

2.5 Stopping Rules

For compatible systems $Ax = b$, an approximate solution x_k may be regarded as acceptable if

$$\|r_k\| \leq (\|A\|\|x_k\| + \|b\|)\mathbf{tol}$$

for some user-specified tolerance (e.g., $\mathbf{tol} = 10^{-4}$, 10^{-8} or 10^{-12} respectively for moderate, accurate, or highly accurate solutions using 15-digit arithmetic). For incompatible (least-squares) problems where $\|r_k\| \not\rightarrow 0$, a good stopping rule is

$$\|Ar_k\| \leq \|A\|\|r_k\|\mathbf{tol}.$$

2.6 Cautions

The methods described above are reliable in practice, *even though V_k will be far from orthogonal*. The work and storage per iteration are constant and minimal ($O(n)$, where n is the dimension of A). The main question remaining is, how many iterations will be required? We hope for far fewer than n iterations, but it could be $5n$ or $10n$ or even more.

If reorthogonalization were used to maintain orthogonal V_k , the iterations would be bounded by n (or more precisely by the number of clusters in the eigenvalue of A). However, all of the vectors v_k would need to be stored, and the work and storage would grow quadratically. We consider this not an option in general.

Many authors present equation (1) correctly, but then derive further results from two equations that don't hold unless full reorthogonalization is used. We emphasize that there is *no need to assume that $V_k^T A V_k = T_k$ and/or $V_k^T b = \beta_1 e_1$, and they quickly cease to be true*. Luckily, equations (1) and (3) are sufficient as they stand.

Similarly, many authors allow an approximate solution x_0 to be provided, and proceed to update the solution inside the solver when it is applied to the system $Ad = r_0$, where $r_0 = b - Ax_0$ and $x = x_0 + d$. Compare the implementations

$$\begin{array}{ll} x \leftarrow x_0 & \text{for } k = 1 : K, \quad x \leftarrow x + \text{correction}, \quad \text{end} \\ d \leftarrow 0 & \text{for } k = 1 : K, \quad d \leftarrow d + \text{correction}, \quad \text{end} \quad x = x_0 + d. \end{array}$$

The first choice is *not recommended* when x_0 is a good approximation, because the x corrections could be small relative to x_0 and many significant digits could be lost. The second choice is safer, at the cost of storing x_0 elsewhere. For this choice, we need to be conscious of solving $Ad = r_0$ when choosing stopping tolerances. If anything is to be gained from x_0 , we need *looser tolerances* than if we were solving $Ax = b$ itself.

2.7 Augmented Systems

The following symmetric system underlies several iterative methods for more general problems:

$$\widehat{A}_{\gamma,\delta}\widehat{x} = \widehat{b} \quad \equiv \quad \begin{pmatrix} \gamma I & A \\ A^T & \delta I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \quad (5)$$

