Stanford University, Management Science & Engineering (and ICME)

MS&E 318 (CME 338) Large-Scale Numerical Optimization

Instructor: Michael Saunders Spring 2018

Notes 5: **Iterative Methods for Square and Rectangular Systems**

1 Introduction

Some unsymmetric or rectangular systems that arise in optimization are:

- Simplex method: $Bx = b$ and $B^T y = c$, where B is a square basis matrix.
- First-order multiplier estimates: $\min_y \|J^T y - g\|$, where J is the Jacobian for the current set of active constraints, and g is the current objective gradient.

Large linear programs usually require the solution of so many square systems that we could not consider using iterative methods to solve $Bx = b$ and $B^T y = c$ within the simplex method. However, special circumstances may arise requiring nonstandard approaches.

For various applications we consider iterative methods for the following problems:

- Square unsymmetric systems: $Ax = b$.
- Under-determined consistent systems: $\min \|x\|^2$ subject to $Ax = b$.
- Over-determined systems (least squares): $\min \|Ax - b\|^2$.
- Regularized systems: $\min \|Ax - b\|^2 + \|\delta x\|^2 \quad \equiv \quad \min \left\| \begin{pmatrix} A \\ \delta I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|^2$, where δ is a scalar and A may have any shape or rank.

These four problems are equivalent to the symmetric system $\begin{pmatrix} \gamma I & A \\ A^T & \delta I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$ with the parameters $(\gamma, \delta) = (0, 0)$, $(0, -1)$, $(1, 0)$, $(\delta, -\delta)$ respectively. They are also equivalent to the definite or semidefinite systems $(A^T A + \gamma^2 I)x = A^T b$ or $(AA^T + \gamma^2 I)y = b$ with $\gamma = 0$ or $-\delta$ respectively, and $x = A^T y$. We may therefore expect the symmetric iterative solvers to apply. However, more effective numerical methods are obtained by working with A directly.

2 Bidiagonalization methods for unsymmetric systems

A square or rectangular matrix A can be reduced to upper bidiagonal form by multiplying alternately on the left and right by certain orthogonal matrices: $U^T A V = B$. This is the starting point for dense singular value decompositions (SVDs) [8]. When A is sparse or a "black box" operator for forming matrix-vector products Av, $A^T u$, the bidiagonalization can be performed iteratively. (Actually, we reduce $\begin{pmatrix} b & A \end{pmatrix}$ to upper bidiagonal form, meaning A is reduced to *lower* bidiagonal form.)

2.1 The Golub-Kahan process

Bidiag$(A, b) \to (B_k, U_{k+1}, V_k)$ or (L_k, U_k, V_k) denotes the following process. Given an $m \times n$ matrix A and a starting vector b, the Golub-Kahan process [6] generates vectors u_k, v_k and positive scalars α_k, β_k $(k = 1, 2, \dots)$ according to these steps:

1. Set $\beta_1 u_1 = b$ and $\alpha_1 v_1 = A^T u_1$. (Exit if $\beta_1 = 0$ or $\alpha_1 = 0$.)
2. For $k = 1, 2, \dots$, set
$$\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k,$$
$$\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k.$$

After k steps, the situation is summarized by the equations

$$AV_k = U_{k+1}B_k = U_kL_k + \beta_{k+1}u_{k+1}e_k^T, \tag{1}$$

$$A^TU_{k+1} = V_kB_k^T + \alpha_{k+1}v_{k+1}e_{k+1}^T = V_{k+1}L_{k+1}^T, \tag{2}$$

where $U_k = \begin{pmatrix} u_1 & u_2 & \dots & u_k \end{pmatrix}$, $V_k = \begin{pmatrix} v_1 & v_2 & \dots & v_k \end{pmatrix}$, L_k is lower bidiagonal, and B_k is also lower bidiagonal with one extra row:

$$L_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{pmatrix}, \qquad B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} = \begin{pmatrix} L_k \\ \beta_{k+1}e_k^T \end{pmatrix}.$$

With exact arithmetic the columns of U_k and V_k would be orthonormal for each k until $\beta_{\ell+1} = 0$ or $\alpha_{\ell+1} = 0$ for some $k = \ell$. In practice, orthonormality is soon lost, but relations (1)–(2) hold to working precision. To retain orthogonality properties, one might guess that each u_{k+1} and v_{k+1} would need to be reorthogonalized with respect to U_k and V_k respectively, at the expense of storing all previous Golub-Kahan vectors. However, Simon and Zha [24] found that one-sided reorthogonalization is enough; that is, if u_{k+1} is reorthogonalized with respect to U_k, then V_k remains essentially orthonormal, and vice versa. The same effect was observed by Fong and Saunders [4]. The Fortran 90 implementation of LSMR allows local or full reorthogonalization of V_k (the shortest vectors for over-determined systems).

A compromise is to use *partial reorthogonalization*, as in Larsen's PROPACK package for computing some of the singular vectors of a sparse matrix or linear operator A [11].

2.2 Properties of the Golub-Kahan process

The vector u_k lies in the Krylov subspace $\mathcal{K}_k(AA^T, b) \equiv \text{span}\{b, AA^Tb, \dots, (AA^T)^{k-1}b\}$, and v_k lies in the Krylov subspace $\mathcal{K}_k(A^TA, A^Tb) \equiv \text{span}\{A^Tb, (A^TA)A^Tb, \dots, (A^TA)^{k-1}A^Tb\}$. Some properties follow:

1. B_k has full column rank k for all $k \leq \ell$.
2. If $\beta_{\ell+1} = 0$, we have $AV_\ell = U_\ell L_\ell$ with L_ℓ nonsingular and $b \in \text{range}(A)$.
3. If $\beta_{\ell+1} > 0$ but $\alpha_{\ell+1} = 0$, we have $AV_\ell = U_{\ell+1}B_\ell$ with $\text{rank}(B_\ell) = \ell$ but $b \notin \text{range}(A)$.

2.3 Golub-Kahan with regularization

Let δ be a given scalar (≥ 0 without loss of generality), and define

$$\widetilde{A} = \begin{pmatrix} A \\ \delta I \end{pmatrix}, \qquad \widetilde{b} = \begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{3}$$

During Bidiag(A, b), orthogonal matrices $\widetilde{Q}_k$ may be constructed from $2k-1$ plane rotations to form the quantities

$$\widetilde{Q}_k \begin{pmatrix} B_k \\ \delta I \end{pmatrix} = \begin{pmatrix} \widetilde{B}_k \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} \widetilde{U}_{k+1} & \widetilde{Y}_k \end{pmatrix} = \begin{pmatrix} U_{k+1} & \\ & V_k \end{pmatrix} \widetilde{Q}_k^T, \tag{4}$$

where $\widetilde{B}_k$ (like B_k) is lower bidiagonal with dimensions $(k+1) \times k$. The following result is obtained straightforwardly from (1)–(4).

Result 1 *If* Bidiag$(A, b) \to (B_k, U_{k+1}, V_k)$, *then* Bidiag$(\widetilde{A}, \widetilde{b}) \to (\widetilde{B}_k, \widetilde{U}_{k+1}, V_k)$.

In short, the bidiagonalization of $\widetilde{A} = \left(\begin{smallmatrix} A \\ \delta I \end{smallmatrix}\right)$ may be obtained efficiently from the bidiagonalization of A itself. The mechanism is less trivial than in the symmetric case. It motivates the subproblems used next.

Table 4: Subproblems defining y_k and $x_k = V_k y_k$ for three algorithms.

Method		Subproblem	Factorization
CRAIG	$\delta = 0$	$L_k y_k = \beta_1 e_1$	
LSQR	$\delta = 0$	$\min \|B_k y_k - \beta_1 e_1\|$	$Q_k B_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix}$
	$\delta > 0$	$\min \left\| \begin{pmatrix} B_k \\ \delta I \end{pmatrix} y_k - \begin{pmatrix} \beta_1 e_1 \\ 0 \end{pmatrix} \right\|$	$Q_k \begin{pmatrix} B_k \\ \delta I \end{pmatrix} = \begin{pmatrix} R_k \\ 0 \\ 0 \end{pmatrix}$
LSMR	$\delta \geq 0$	$\min \left\| \begin{pmatrix} R_k^T R_k \\ \bar{\beta}_{k+1} e_k^T \end{pmatrix} y_k - \bar{\beta}_1 e_1 \right\|$	$\bar{Q}_{k+1} \begin{pmatrix} R_k^T \\ \varphi_k e_k^T \end{pmatrix} = \begin{pmatrix} \bar{R}_k \\ 0 \end{pmatrix}$

Table 5: Definition of W_k and z_k such that $x_k = V_k y_k = W_k z_k$.

		W_k	z_k
CRAIG	$\delta = 0$	V_k	$z_k = y_k$
LSQR	$\delta \geq 0$	$V_k R_k^{-1}$	$Q_k \beta_1 e_1 = \begin{pmatrix} z_k \\ \bar{\zeta}_{k+1} \end{pmatrix}$
LSMR	$\delta \geq 0$	$V_k R_k^{-1} \bar{R}_k^{-1}$	$\bar{Q}_k \beta_1 e_1 = \begin{pmatrix} z_k \\ \bar{\zeta}_{k+1} \end{pmatrix}$

2.4 CRAIG, LSQR, and LSMR

As in the symmetric algorithms, we can use various subproblems to define vectors y_k and solution estimates $x_k = V_k y_k$. Craig's method [3] (as derived by Paige [15]) is the simplest method for solving unsymmetric $Ax = b$, but not least-squares problems. LSQR [16, 17] and LSMR [4] apply to both consistent and inconsistent systems. Table 4 shows the subproblems and the factorizations needed to solve them. Table 5 shows how the factorizations are used to obtain estimates $x_k = W_k z_k$ that permit updates: $x_k = x_{k-1} + \zeta_k w_k$.

From (1) we have

$$\begin{aligned} r_k \equiv b - Ax_k &= \beta_1 u_1 - AV_k y_k \\ &= U_{k+1}(\beta_1 e_1 - B_k y_k) = U_{k+1} t_{k+1}, \qquad (5) \\ \text{where} \quad t_{k+1} &\equiv \beta_1 e_1 - B_k y_k. \qquad (6) \end{aligned}$$

Since $\|U_{k+1}\| = O(1)$, we would like t_{k+1} to be small (when $\delta = 0$). Craig's method makes $t_{k+1} = 0$ everywhere except its last element, giving $r_k = -\eta_k \beta_{k+1} u_{k+1}$ (where η_k is the last element of y_k), while LSQR is more balanced in minimizing $\|t_{k+1}\|$ (which has the effect of minimizing $\|r_k\|$).

From (2) and (5)–(6) we have

$$\begin{aligned} A^T r_k = A^T U_{k+1} t_{k+1} &= V_{k+1} L_{k+1}^T t_{k+1} \\ &= V_{k+1} \left[\begin{pmatrix} \alpha_1 \beta_1 \\ 0 \end{pmatrix} - \begin{pmatrix} B_k^T B_k \\ \alpha_{k+1} \beta_{k+1} e_k^T \end{pmatrix} y_k \right]. \qquad (7) \end{aligned}$$

With $B_k^T B_k = R_k^T R_k$ and $\bar{\beta}_k \equiv \alpha_k \beta_k$, we see that the LSMR subproblem minimizes $\|A^T r_k\|$. The LSMR factorization needs $\varphi_k = \bar{\beta}_{k+1}/\rho_k$, where ρ_k is the last diagonal of R_k.

Result 2 *CRAIG is equivalent to CG on $AA^T y = b$, where $x = A^T y$.*
LSQR is equivalent to CG on the normal equation $(A^T A + \delta^2 I)x = A^T b$.
LSMR is equivalent to MINRES on the normal equation.

The CRAIG iteration Given the Golub-Kahan process, CRAIG needs no further factorizations. Forward substitution on $L_k y_k = \beta_1 e_1$ gives $\eta_1 = \beta_1/\alpha_1$, $\eta_2 = -\beta_2\eta_1/\alpha_2$, ..., and we have theoretically orthogonal updates to $x_0 \equiv 0$:

$$\eta_k = -\beta_k \eta_{k-1}/\alpha_k, \qquad x_k = V_k y_k = x_{k-1} + \eta_k v_k.$$

The LSQR iteration The subproblem that allows LSQR to incorporate regularization was first proposed by Björck [1]. Result 1 helps confirm that the resulting method is equivalent to applying LSQR to $\widetilde{A}$ and $\widetilde{b}$. (Working backwards, the proof of Result 1 reveals the need for the orthogonal factorization (4), which in turn suggests the subproblem.)

The QR factorizations for LSQR can be computed at negligible cost using one plane rotation for each k when $\delta = 0$, or two rotations when $\delta > 0$, giving upper-bidiagonal factors R_k. The matrix Q_k, a product of the rotations, does not need to be saved. The columns of W_k are obtained by forward substitution: $R_k^T W_k^T = V_k^T$. Then with $x_0 \equiv 0$,

$$x_k = V_k y_k = W_k R_k y_k = W_k z_k = x_{k-1} + \zeta_k w_k.$$

The LSMR iteration For LSMR we define $t_k = R_k y_k$ and solve $R_k^T q_k = \bar{\beta}_{k+1} e_k$ to get $q_k = (\bar{\beta}_{k+1}/\rho_k)e_k = \varphi_k e_k$ with $\rho_k = (R_k)_{kk}$ and $\varphi_k \equiv \bar{\beta}_{k+1}/\rho_k$. Then we perform a second QR factorization

$$\bar{Q}_{k+1} \begin{pmatrix} R_k^T & \bar{\beta}_1 e_1 \\ \varphi_k e_k^T & 0 \end{pmatrix} = \begin{pmatrix} \bar{R}_k & z_k \\ 0 & \bar{\zeta}_{k+1} \end{pmatrix}, \qquad \bar{R}_k = \begin{pmatrix} \bar{\rho}_1 & \bar{\theta}_2 & & \\ & \bar{\rho}_2 & \ddots & \\ & & \ddots & \bar{\theta}_k \\ & & & \bar{\rho}_k \end{pmatrix}. \tag{8}$$

Combining with (7) gives

$$\begin{aligned} \min_{y_k} \|A^T r_k\| = \min_{y_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} R_k^T R_k \\ q_k^T R_k \end{pmatrix} y_k \right\| &= \min_{t_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} R_k^T \\ \varphi_k e_k^T \end{pmatrix} t_k \right\| \\ &= \min_{t_k} \left\| \begin{pmatrix} z_k \\ \bar{\zeta}_{k+1} \end{pmatrix} - \begin{pmatrix} \bar{R}_k \\ 0 \end{pmatrix} t_k \right\|. \end{aligned} \tag{9}$$

The subproblem is solved by choosing t_k from $\bar{R}_k t_k = z_k$. Let W_k and $\bar{W}_k$ be computed by forward substitution from $R_k^T W_k^T = V_k^T$ and $\bar{R}_k^T \bar{W}_k^T = W_k^T$. Then from $x_k = V_k y_k$, $R_k y_k = t_k$, and $\bar{R}_k t_k = z_k$, we have $x_0 \equiv 0$ and

$$x_k = W_k R_k y_k = W_k t_k = \bar{W}_k \bar{R}_k t_k = \bar{W}_k z_k = x_{k-1} + \zeta_k \bar{w}_k.$$

The following table compares the costs.

	Storage	Work per iteration
CRAIG, $\delta = 0$	$m + 2n$	$3m + 4n$
LSQR, any δ	$m + 3n$	$3m + 5n$
LSMR, any δ	$m + 4n$	$3m + 6n$

2.5 Preferences

Let $r_k = b - Ax_k$ be the residual and $d_k = x - x_k$ be the corresponding error. The properties of CRAIG are similar to those of SYMMLQ. The CRAIG point solves both of the problems

$$\begin{aligned} &\min_{y_k} \|d_k\| \quad \text{such that} \quad x_k = V_k y_k, \\ &\min_{y_k} \|x_k\| \quad \text{such that} \quad x_k = V_k y_k,\ U_k^T r_k = 0, \end{aligned}$$

so that $\|d_k\|$ decreases, $\|x_k\|$ increases, and the system must be consistent. LSQR chooses y_k to solve the problem

$$\min_{y_k} \|r_k\| \quad \text{such that} \quad x_k = V_k y_k,$$

so that $\|r_k\|$ decreases and the system may be inconsistent ($\|r_k\| \not\to 0$). LSMR chooses y_k to solve

$$\min_{y_k} \|A^T r_k\| \quad \text{such that} \quad x_k = V_k y_k,$$

so that $\|A^T r_k\|$ decreases, and again the system may be inconsistent.

Further discussion is given in [22, 4]. While CRAIG is slightly more economical, a conclusion in [22] is that LSQR is suitable for all cases in the sense that it is reliable on square, over-determined, and under-determined systems, with or without regularization. We expect the same for LSMR. For consistent systems, LSQR is probably best because it minimizes $\|r_k\|$. For least-squares problems, LSMR seems preferable if iterations are terminated early, because it is $\|A^T r_k\|$ that must decrease to zero. A fortunate surprise is that LSMR's $\|r_k\|$ also decreases, and is never far behind $\|r_k\|$ for LSQR.

Note that the solutions $x_k = V_k y_k$ lie in the Krylov subspace $\mathcal{K}_k(A^T A, A^T b)$. The convergence of these bidiagonalization-based methods depends on the eigenvalues of $A^T A$ (the squares of the singular values of A).

2.6 Estimation of norms

At iteration k of LSQR, estimates of $\|r_k\|$, $\|A^T r_k\|$, $\|x_k\|$, $\|A\|$, and $\text{cond}(A)$ can be obtained at minimal cost. All five items are used in LSQR's stopping rules. To estimate norms, it is often necessary to assume that the columns of U_k and V_k are orthonormal ($U_k^T U_k = I$, $V_k^T V_k = I$). Although this is rarely true in practice, the resulting estimates have proved to be remarkably reliable.

For simplicity we assume $\delta = 0$. As shown in [16], the following relations can be derived from (1), (2), (5), (6), and the QR factorization of B_k in Table 4:

$$\begin{aligned} r_k &= \bar{\zeta}_{k+1} U_{k+1} Q_k^T e_{k+1} \\ \|r_k\| &= \bar{\zeta}_{k+1} \;=\; \beta_1 s_1 s_2 \dots s_k \\ A^T r_k &= -(\bar{\zeta}_{k+1} \alpha_{k+1} c_k) v_{k+1} \\ \|A^T r_k\| &= \bar{\zeta}_{k+1} \alpha_{k+1} |c_k|. \end{aligned}$$

With orthogonality assumptions we also have $V_k^T A^T A V_k = B_k^T B_k = R_k^T R_k$, and so from the Courant-Fischer minimax theorem, the eigenvalues of $B_k^T B_k$ are bounded above and below by the largest and smallest nonzero eigenvalues of $A^T A$. The same can be said of the singular values of B_k compared to those of A. It follows for the 2- and F-norms that $\|B_k\| \le \|A\|$ and $\|R_k^{-1}\| = \|B_k^+\| \le \|A^+\|$. With $W_k = V_k R_k^{-1}$ we now have

$$1 \le \|B_k\| \|W_k\| \le \|A\| \|A^+\| = \text{cond}(A)$$

for the 2- and F-norms. Hence we use the monotonically increasing estimates

$$\|A\|_F \approx \|B_k\|_F, \qquad \text{cond}(A) \approx \|B_k\|_F \|W_k\|_F,$$

where $\|B_k\|_F^2 = \|B_{k-1}\|_F^2 + \alpha_k^2 + \beta_{k+1}^2$ and $\|W_k\|_F^2 = \|W_{k-1}\|_F^2 + \|w_k\|^2$ can be updated easily.

To estimate $\|x_k\|$, we make use of the so-called QLP factorization of B_k (see Stewart [28]). The QR factorization of B_k can be followed by an orthogonal transformation from the right that reduces R_k to lower-bidiagonal form: $R_k P_k = \bar{L}_k$. Recall that the least-squares subproblem for defining y_k is solved by $R_k y_k = z_k$. If we define $y_k = P_k p_k$ for some vector

p_k, we have $R_k y_k = R_k P_k p_k = \bar{L}_k p_k = z_k$, where p_k is obtained by forward substitution and hence differs from p_{k-1} in just its last element π_k. We now have $\|x_k\|^2 = \|V_k y_k\|^2 \approx \|y_k\|^2 = \|p_k\|^2$ and hence

$$\|x_k\|^2 = \|p_{k-1}\|^2 + \pi_k^2 = \|x_{k-1}\|^2 + \pi_k^2.$$

This construction shows that $\|x_k\|$ increases monotonically for LSQR. The same approach applies to MINRES, and has been implemented in MINRES-QLP [2].

For LSMR, estimates of $\|r_k\|$, $\|A^T r\|$, $\|x_k\|$, $\|A\|$, and cond(A) can also be obtained at negligible cost (although it is simpler to compute $\|x_k\|$ directly).

2.7 Stopping rules

We formulate rules for terminating LSQR and LSMR in terms of three dimensionless quantities α, β, γ specified by a user. (They are called `Atol`, `btol`, `conlim` in the software.) The first two rules apply to consistent and inconsistent systems respectively. The third rule applies to both.

S1 Stop if $\|r_k\| \le \psi_k \equiv \alpha\|A\|\|x_k\| + \beta\|b\|$.

S2 Stop if $\dfrac{\|A^T r_k\|}{\|A\|\,\|r_k\|} \le \alpha$.

S3 Stop if cond$(A) \ge \gamma$.

Rules S1 and S2 are based on allowable perturbations in the data. The user may therefore set α and β according to the (known or estimated) accuracy of the data. Rule S3 represents an attempt to regularize ill-conditioned systems.

To justify S1, note that x_k is the exact solution of the perturbed system

$$(A + E_k)x_k = b + f_k, \qquad E_k = \frac{\alpha\|A\|}{\psi_k\|x_k\|} r_k x_k^T, \qquad \|E_k\| = \alpha\|A\|\frac{\|r_k\|}{\psi_k},$$
$$f_k = -\frac{\beta\|b\|}{\psi_k} r_k, \qquad \|f_k\| = \beta\|b\|\frac{\|r_k\|}{\psi_k}$$

for any nonnegative α and β. If S1 is satisfied ($\|r_k\| \le \psi_k$) we see that x_k is an "acceptable solution" in the sense that $\|E_k\|/\|A\| \le \alpha$ and $\|f_k\|/\|b\| \le \beta$. Titley-Peloquin [29] shows that these E_k and f_k are the *smallest* perturbations to A and b that make x_k acceptable.

To justify S2, Stewart [27] noted that x_k and $\widetilde{r}_k \equiv b - (A + E_k)x_k = r_k - E_k x_k$ are the exact solution and residual for the perturbed least-squares problem

$$\min_x \|(A + E_k)x - b\|, \qquad E_k \equiv -\frac{r_k r_k^T A}{\|r_k\|^2}, \qquad \|E_k\| = \|A^T r_k\|/\|r_k\|.$$

Hence the perturbation to A is sufficiently small (x_k is an acceptable solution) if S2 is satisfied. This perturbation E_k is not the smallest possible, but it is cheaply computable. We do know analytically the smallest perturbations to A and b that make x_k an exact least-squares solution (Higham [10, pp. 392–393]), but normally they are much too expensive to evaluate.

Stopping rule S3 is based on the following arguments. Suppose that $A = USV^T$ (the SVD) with $S = \text{diag}(\sigma_j)$, $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n \ge 0$. As k increases, our estimate cond$(A) \approx \|B_k\|_F\|W_k\|_F$ tends to level off near the values of the ordered sequence σ_1/σ_1, σ_1/σ_2, σ_1/σ_3, $\ldots$, with varying numbers of iterations near each level. This suggests rule S3 as a means of regularizing ill-conditioned problems, as in the discretization of ill-posed problems (e.g., [9]). For example, if the singular values of A were known to be of order 1, 0.9, 10^{-3}, 10^{-6}, 10^{-7}, the effect of the two smallest singular values could be suppressed by setting $\gamma = 10^4$.

3 Arnoldi-based methods for square systems

For square unsymmetric systems $Ax = b$, it may seem desirable to develop a method in which the approximate solutions $x_k = V_k y_k$ lie in the Krylov subspace $\mathcal{K}_k(A, b)$ rather than $\mathcal{K}_k(A^TA, A^Tb)$ for LSQR and LSMR. Indeed this is possible, but at the cost of steadily increasing work and storage.

3.1 The Arnoldi process

Given a general $n \times n$ matrix A and a starting vector b, the Arnoldi process generates vectors v_k and scalars β_k and h_{ik} as follows:

1. Set $\beta_1 v_1 = b$. (Exit if $\beta_1 = 0$.)
2. For $k = 1, 2, \ldots, n$
   Compute $w = Av_k$
   For $i = 1, 2, \ldots, k$, set $h_{ik} = v_i^T w$, $w = w - h_{ik} v_i$.
   Set $\beta_{k+1} v_{k+1} = w$. (If $\beta_{k+1} = 0$, define $\ell = k$ and exit.)

This is called the Modified Gram-Schmidt version of Arnoldi, and leads to the MGS-GMRES algorithm for solving $Ax = b$. (The Arnoldi process is used by ARPACK [12] to compute some eigenvalues/eigenvectors of large matrices.) As for the symmetric Lanczos process, each matrix $V_k = \begin{pmatrix} v_1 & v_2 & \ldots & v_k \end{pmatrix}$ would have orthonormal columns with exact arithmetic. After k steps, the situation may be summarized as

$$AV_k = V_{k+1} H_k, \tag{10}$$

where H_k is now a $(k+1) \times k$ Hessenberg matrix

$$H_k = \begin{pmatrix} h_{11} & h_{12} & \ldots & \ldots & h_{1k} \\ \beta_2 & h_{22} & \ldots & \ldots & h_{2k} \\ & \beta_3 & \ldots & \ldots & h_{3k} \\ & & \ddots & \vdots & \vdots \\ & & & \beta_k & h_{kk} \\ & & & & \beta_{k+1} \end{pmatrix}.$$

The process stops with $k = \ell$ and $\beta_{\ell+1} = 0$ for some $\ell \leq n$, and then $AV_\ell = V_\ell T_\ell$, where T_ℓ is upper Hessenberg and square.

3.2 GMRES

If we define $x_k = V_k y_k$ for some y_k, (10) gives the residual

$$r_k \equiv b - Ax_k = \beta_1 v_1 - AV_k y_k = V_{k+1}(\beta_1 e_1 - H_k y_k), \tag{11}$$

which suggests the least-squares subproblem $\min \|H_k y_k - \beta_1 e_1\|$ for defining y_k (as in MINRES and LSQR). This is the basis for the *generalized minimum residual* method GMRES of Saad and Schultz [20]. For each k, a single plane rotation maintains the QR factorization

$$Q_k H_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix}, \qquad Q_k(\beta_1 e_1) = \begin{pmatrix} z_k \\ \bar{\zeta}_{k+1} \end{pmatrix}, \tag{12}$$

from which the least-squares solution can be obtained when necessary by back-substitution: $R_k y_k = z_k$. Before that we have

$$r_k = V_{k+1} Q_k^T \left(\begin{pmatrix} z_k \\ \bar{\zeta}_{k+1} \end{pmatrix} - \begin{pmatrix} R_k \\ 0 \end{pmatrix} y_k \right) = \bar{\zeta}_{k+1} V_{k+1} Q_k^T e_{k+1},$$

so the estimate $\|r_k\| \approx |\bar{\zeta}_{k+1}|$ is available without y_k or x_k (and as for MINRES and LSQR it is a reliable estimate in practice, even if the columns of V_k are not reorthogonalized).

Note that the Arnoldi process requires all columns of V_k to be retained. Whenever $\|r_k\|$ is judged suitably small (or when k reaches a pre-defined limit m), the simplest form of GMRES terminates by solving $R_k y_k = z_k$ and forming $x_k = V_k y_k$ directly. There is no need to form $x_1, x_2, \ldots, x_{k-1}$.

3.3 GMRES(m) = restarted GMRES

For large systems, the iteration limit m is necessarily quite small (perhaps only 10 or 20). The simplest approach is to build a loop around GMRES analogous to iterative refinement for linear systems. The resulting algorithm is called GMRES(m):

1. Given x_0, form $r = b - Ax_0$.
2. If $\|r\|$ is sufficiently small, exit with $x = x_0$.
3. Run the Arnoldi process for m iterations, starting with $\beta_1 v_1 = r$.
4. Solve the least-squares problem $\min \|H_m y_m - \beta_1 e_1\|$ and form $\Delta x = V_m y_m$.
5. Set $x_0 \leftarrow x_0 + \Delta x$ and return to step 1.

A difficulty with GMRES(m) is the need to choose m.

If A is positive definite (the symmetric part $(A + A^T)/2$ is positive definite), it can be proved that GMRES(m) converges for any $m \geq 1$ [19].

If A is diagonalizable—i.e., it has a full set of eigenvalues and can be written as $A = XDX^{-1}$ with D diagonal (but perhaps complex)—the *rate of convergence* can be bounded in terms of the condition number of X. This is most useful if A is normal or nearly normal ($A^TA \approx AA^T$), in which case $\text{cond}(X) \approx 1$.

For more general matrices A, GMRES is in danger of *stagnation*, in the sense that progress may become negligible for any practical value of m. (This is in contrast to CRAIG, LSQR, and LSMR, which may converge slowly at times but will *always get there* if allowed to iterate long enough, and they do so using a constant amount of storage. The same is true of the symmetric methods CG, SYMMLQ, MINRES.) Nevertheless, GMRES is a remarkably popular method for solving unsymmetric square $Ax = b$. One advantage is that GMRES uses products Av but no transpose products A^Tu (which may not be available). A further saving grace is the availability of effective preconditioners for many practical cases.

3.4 DQGMRES(m)

Another way to limit the storage needed for GMRES was proposed by Saad and Wu [21]. DQGMRES(m) limits the inner loop of the Arnoldi process to be

$$\text{For } i = \max\{1, k-m+1\}, \ldots, k, \text{ set } h_{ik} = v_i^T w, \; w = w - h_{ik} v_i.$$

The resulting H_k is upper Hessenberg with a bandwidth of $m+1$. For example, $m = 2$ gives

$$H_k = \begin{pmatrix} h_{11} & h_{12} & & & \\ \beta_2 & h_{22} & h_{23} & & \\ & \beta_3 & h_{33} & h_{34} & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_k & h_{kk} \\ & & & & \beta_{k+1} \end{pmatrix}.$$

The name DQGMRES means "direct quasi-GMRES". Equations (10)-(11) still hold, but V_k no longer has orthonormal columns for $k > m$. The QR factorization (12) gives an upper-triangular R_k with bandwidth $m + 1$, and it is efficient to proceed exactly as in MINRES: solve $R_k^T W_k^T = V_k^T$ and update $x_k = x_{k-1} + \zeta_k w_k$. This method deserves more attention!

4 Preconditioning

It is a cliche to say that iterative methods need good preconditioners, but this is the main hope for minimizing the total iterations required, and for keeping m low for GMRES(m) and DQGMRES(m).

For square systems $Ax = b$, there is a choice of left, right, or split preconditioning. We need matrices C_1, C_2 such that $C_1C_2 \approx A$ and systems $C_iz = v$ and possibly $C_i^Tz = u$ can be solved efficiently ($i = 1, 2$). The iterative method is then applied to $C_1^{-1}AC_2^{-1}y = C_1^{-1}b$, and the solution is recovered by solving $C_2x = y$. The choice of C_i inevitably depends on each application.

In the absence of other knowledge, the *very least* we should do is apply row and column scaling. The general aim would be to ensure that all rows and all columns have essentially the same norm. Some efficient scaling algorithms are described in [13] for symmetric, square, and rectangular systems.

More generally, incomplete Cholesky or LU factorizations of A are commonly used, in which "incomplete" factors are more sparse than exact factors.

For least-squares problems, we must use $C_1 = I$ to avoid altering the problem. Again, the *very least* we should do to help LSQR and LSMR is apply column scaling to make the 2-norms of all columns of AC_2^{-1} equal. Thus, C_2 should be a diagonal matrix whose jth diagonal is $\|a_j\|_2$, where a_j is the corresponding column of A. (This is *diagonal preconditioning.*)

More generally, C_2 should approximate the R part of a QR factorization of A (because the exact R would give convergence in one iteration—the perfect preconditioner). It is more likely that we could compute sparse rectangular LU factors $P_1AP_2 = \begin{pmatrix} L_1 & \\ L_2 & I \end{pmatrix} \begin{pmatrix} U \\ 0 \end{pmatrix}$ with the L factor well-conditioned, and then $C_2 = UP_2^T$ may be effective.

5 Other methods for unsymmetric systems

LSQR has been the primary iterative solver for over-determined (least-squares) problems, but LSMR will probably begin to take over because it can stop sooner. Both solvers apply to square and under-determined systems $Ax = b$ (for which LSQR can stop sooner).

Other important methods for square unsymmetric systems are USYMLQ and USYMQR [23], CGS [25], QMR [5], BiCGSTAB [30], and IDR(s) [26]. Some intriguing test cases are given by Nachtigal et al. [14] to show that LSQR, GMRES, and CGS are fundamentally different methods that can outperform each other (in iteration counts) by factors as large as $\sqrt{n}$ or n.

USYMLQ and USYMQR [23] are based on an orthogonal tridiagonalization that becomes the symmetric Lanczos process when the matrix involved is symmetric. Two starting vectors b and c and products Av_k and A^Tu_k are used to generate two orthonormal sets of vectors $V_k = \begin{pmatrix} v_1 \dots v_k \end{pmatrix}$ and $U_k = \begin{pmatrix} u_1 \dots u_k \end{pmatrix}$ for which

$$\begin{aligned} \begin{pmatrix} b & AV_k \end{pmatrix} &= U_{k+1} \begin{pmatrix} \beta e_1 & T_{k+1,k} \end{pmatrix}, \\ \begin{pmatrix} c & A^TU_k \end{pmatrix} &= V_{k+1} \begin{pmatrix} \gamma e_1 & T_{k,k+1}^T \end{pmatrix}, \end{aligned}$$

where $T_{p,q} \in \mathbb{R}^{p \times q}$ is tridiagonal. Two systems $Ax = b$ and $A^Ty = c$ can then be solved simultaneously. Reichel and Ye [18] rediscovered the tridiagonalization and recognized that it applies equally well to *rectangular* systems. With square and rectangular systems in mind they named their solver GLSQR, even though it does not reduce to LSQR in any particular case. They showed that careful choice of c (namely $c \approx x$) can give good approximate solutions to $Ax \approx b$ in fewer iterations than LSQR.

Soon after, Golub, Stoll, and Wathen [7] used the orthogonal tridiagonalization to estimate the "scattering amplitude" $c^Tx = b^Ty$ without computing x and y.

5.1 Popularity contest

It is interesting to judge how frequently each Krylov solver finds application, as estimated by the number of references in the literature. Searches using Google Scholar for various sets of keywords

```
  cg      hestenes stiefel       gmres    saad schultz          idr  sonneveld "van Gijzen"
  symmlq  paige saunders         qmr      freund nachtigal      lsqr paige saunders
 "minres" paige saunders         bicgstab "van der Vorst"       lsmr fong  saunders
```

have given the following results in recent years. Before 2016, the MINRES results were evidently exaggerated because `minres` without quotes was interpreted as `miners` and included references about lung cancer(!). Later searches for `"minres"` are more realistic.

April	CG	SYMMLQ	MINRES	GMRES	QMR	BiCGSTAB	IDR(s)	LSQR	LSMR
2011	1990	310	16100	5310	1080	1720	88	1780	6
2012	2200	324	16300	5480	1100	1870	117	1920	18
2013	2620	372	16700	6390	1220	2080	161	2390	38
2014	3070	387	2080	6860	1270	2270	168	2550	758
2016	3390	447	919	8010	1370	2670	236	3190	527
2017	3950	479	1040	9050	1430	3060	300	3660	607
2018	3970	497	1140	9100	1450	3170	295	3850	736

References

[1] Å. Björck. A bidiagonalization algorithm for solving ill-posed systems of linear equations. Report LITH-MAT-R-80-33, Dept. of Mathematics, Linköping University, Linköping, Sweden, 1980.

[2] S.-C. Choi. *Iterative Methods for Singular Linear Equations and Least-Squares Problems*. PhD thesis, ICME, Stanford University, Dec 2006.

[3] J. Craig. The n-step iteration procedure. *J. Math. and Phys.*, 34:64–73, 1955.

[4] D. C.-L. Fong and M. A. Saunders. LSMR: An iterative algorithm for least-squares problems. *SIAM J. Sci. Comput.*, 33(5):2950–2971, 2011. `http://stanford.edu/group/SOL/software.html`.

[5] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. In R. Beauwens and P. de Groen, editors, *Iterative Methods in Linear Algebra*, pages 151–154. Elsevier Science Publishers, 1992.

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

[7] G. H. Golub, M. Stoll, and A. Wathen. Approximation of the scattering amplitude and linear systems. *ETNA*, 31:178–203, 2008.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, fourth edition, 2013.

[9] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, Philadelphia, 1998.

[10] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, second edition, 2002.

[11] R. M. Larsen. PROPACK software for SVD of sparse matrices. `http://soi.stanford.edu/~rmunk/PROPACK/`, 2005.

[12] R. Lehoucq, K. Maschhoff, D. Sorensen, and C. Yang. ARPACK software for EVD of sparse matrices. `http://www.caam.rice.edu/software/ARPACK/`, 1997–present.

[13] Oren E. Livne and Gene H. Golub. Scaling by binormalization. *Numer. Algor.*, 35(1):97–120, 2004.

[14] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM J. Matrix Anal. Appl.*, 13(3):778–795, 1992.

[15] C. C. Paige. Bidiagonalization of matrices and solution of linear equations. *SIAM J. Numer. Anal.*, 11:197–209, 1974.

[16] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71, 1982.

[17] C. C. Paige and M. A. Saunders. Algorithm 583; LSQR: Sparse linear equations and least-squares problems. *ACM Trans. Math. Software*, 8(2):195–209, 1982.

[18] L. Reichel and Q. Ye. A generalized LSQR algorithm. *Numer. Linear Algebra Appl.*, 15:643–660, 2008.

[19] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, second edition, 2003.

[20] Y. Saad and M. H. Schultz. GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. and Statist. Comput.*, 7:856–869, 1986.

[21] Y. Saad and K. Wu. DQGMRES: a direct quasi-minimal residual algorithm based on incomplete orthogonalization. *Numer. Linear Algebra Appl.*, 3(4):329–343, 1996. A copy from the Stanford library is here: `http://stanford.edu/class/msande318/refs/DQGMRES.pdf`.

[22] M. A. Saunders. Solution of sparse rectangular systems using LSQR and CRAIG. *BIT Numer. Math.*, 35:588–604, 1995.

[23] M. A. Saunders, H. D. Simon, and E. L. Yip. Two conjugate-gradient-type methods for unsymmetric linear equations. *SIAM J. Numer. Anal.*, 25(4):927–940, 1988.

[24] H. D. Simon and H. Zha. Low-rank matrix approximation using the Lanczos bidiagonalization process with applications. *SIAM J. Sci. Comput.*, 21(6):2257–2274, 2000.

[25] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. and Statist. Comput.*, 10(1):36–52, 1989.

[26] P. Sonneveld and M. B. van Gijzen. IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM J. Sci. Comput.*, 31(2):1035–1062, 2008.

[27] G. W. Stewart. Research, development and LINPACK. In J. R. Rice, editor, *Mathematical Software III*, pages 1–14. Academic Press, New York, 1977.

[28] G. W. Stewart. The QLP approximation to the singular value decomposition. *SIAM J. Sci. Comput.*, 20(4):1336–1348, 1999.

[29] D. Titley-Peloquin. *Backward Perturbation Analysis of Least Squares Problems*. PhD thesis, School of Computer Science, McGill University, 2010.

[30] 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. and Statist. Comput.*, 13(2):631–644, 1992.