# ALGORITHM & DOCUMENTATION: MINRES-QLP for Symmetric and Hermitian Linear Equations and Least-Squares Problems

SOU-CHENG T. CHOI
University of Chicago/Argonne National Laboratory
and
MICHAEL A. SAUNDERS
Stanford University

---

We describe algorithm MINRES-QLP and its FORTRAN 90 implementation for solving symmetric or Hermitian linear systems or least-squares problems. If the system is singular, MINRES-QLP computes the unique minimum-length solution (also known as the pseudoinverse solution), which generally eludes MINRES. In all cases, it overcomes a potential instability in the original MINRES algorithm. A positive-definite preconditioner may be supplied. Our FORTRAN 90 implementation illustrates a design pattern that allows users to make problem data known to the solver but hidden and secure from other program units. In particular, we circumvent the need for reverse communication. Example test programs input and solve real or complex problems specified in Matrix Market format. While we focus here on a FORTRAN 90 implementation, we also provide and maintain MATLAB versions of MINRES and MINRES-QLP.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems (direct and iterative methods)*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*statistical computing; statistical software*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN program units*

General Terms: Algorithms

Additional Key Words and Phrases: Krylov subspace method, Lanczos process, conjugate-gradient method, singular least-squares, linear equations, minimum-residual method, pseudoinverse solution, ill-posed problem, regression, sparse matrix, data encapsulation

---

Revised October 1, 2013. MINRESQLP_DOC24.pdf.

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, US Department of Energy contract DE-AC02-06CH11357; National Science Foundation grant CCR-0306662; Office of Naval Research grants N00014-02-1-0076 and N00014-08-1-0191; US Army Research Laboratory through the Army High Performance Computing Research Center, DOE through the Scientic Discovery through Advanced Computing program, grant DE-FG02-09ER25917, and NIH award 1U01GM102098-01.
Authors' addresses: S.-C. T. Choi, Computation Institute, University of Chicago, Chicago, IL 60637; email: sctchoi@uchicago.edu; M. A. Saunders, Department of Management Science and Engineering, Stanford University, Stanford, CA 94305-4121; email: saunders@stanford.edu.

Argonne National Laboratory Preprint ANL/MCS-P3027-0812.

## 1. INTRODUCTION

MINRES-QLP [Choi 2006; Choi et al. 2011] is a Krylov subspace method for computing the minimum-length and minimum-residual solution (also known as the pseudoinverse solution) $x$ to the following linear systems or least-squares (LS) problems:

$$\text{solve } Ax = b, \tag{1}$$

$$\text{minimize } \|x\|_2 \quad \text{s.t.} \quad Ax = b, \tag{2}$$

$$\text{minimize } \|x\|_2 \quad \text{s.t.} \quad x \in \arg\min_x \|Ax - b\|_2, \tag{3}$$

where $A$ is an $n \times n$ symmetric or Hermitian matrix and $b$ is a real or complex $n$-vector. Problems (1) and (2) are treated as special cases of (3). The matrix $A$ is usually large and sparse, and it may be singular.[1] It is defined by means of a user-written subroutine `Aprod`, whose function is to compute the product $y = Av$ for any given vector $v$.

Let $x_k$ be the solution estimate associated with MINRES-QLP's $k$th iteration, with residual vector $r_k = b - Ax_k$. Without loss of generality, we define $x_0 = 0$. MINRES-QLP provides recurrent estimates of $\|x_k\|$, $\|r_k\|$, $\|Ar_k\|$, $\|A\|$, $\text{cond}(A)$, and $\|Ax_k\|$, which are used in the stopping conditions.

Other iterative methods specialized for symmetric systems $Ax = b$ are the conjugate-gradient method (CG) [Hestenes and Stiefel 1952], SYMMLQ and MINRES [Paige and Saunders 1975], and SQMR [Freund and Nachtigal 1994]. Each method requires one product $Av_k$ at each iteration for some vector $v_k$. CG is intended for positive-definite $A$, whereas the other solvers allow $A$ to be indefinite.

If $A$ is singular, SYMMLQ requires the system to be consistent, whereas MINRES returns an LS solution for (3) but generally not the min-length solution; see [Choi 2006; Choi et al. 2011] for examples. SQMR without preconditioning is mathematically equivalent to MINRES but could fail on a singular problem. To date, MINRES-QLP is probably the most suitable CG-type method for solving (3).

In some cases the more established symmetric methods may still be preferable.

(1) If $A$ is positive definite, CG minimizes the energy norm of the error $\|x - x_k\|_A$ in each Krylov subspace and requires slightly less work per iteration. However, CG, MINRES, and MINRES-QLP do reduce $\|x - x_k\|_A$ and $\|x - x_k\|$ monotonically. Also, MINRES and MINRES-QLP often reduce $\|r_k\|$ to the desired level significantly sooner than does CG, and the backward error for each $x_k$ decreases monotonically. (See Section 2.4 and [Fong 2011; Fong and Saunders 2012].)

(2) If $A$ is indefinite but $Ax = b$ is consistent (e.g., if $A$ is nonsingular), SYMMLQ requires slightly less work per iteration, and it reduces the error norm $\|x - x_k\|$ monotonically. MINRES and MINRES-QLP *usually* reduce $\|x - x_k\|$ [Fong 2011; Fong and Saunders 2012].

(3) If $A$ is indefinite and well-conditioned and $Ax = b$ is consistent, MINRES might be preferable to MINRES-QLP because it requires the same number of iterations but slightly less work per iteration.

[1] A further input parameter $\sigma$ (a real shift parameter) causes MINRES-QLP to treat "$A$" as if it were $A - \sigma I$. For example, "singular $A$" really means that $A - \sigma I$ is singular.

Argonne National Laboratory Preprint ANL/MCS-P3027-0812.

(4) MINRES and MINRES-QLP require a preconditioner to be positive definite. SQMR might be preferred if $A$ is indefinite and an effective indefinite preconditioner is available.

MINRES-QLP has two phases. Iterations start in the *MINRES phase* and transfer to the *MINRES-QLP phase* when a subproblem (see (8) below) becomes ill-conditioned by a certain measure. If every subproblem is of full rank and well-conditioned, the problem can be solved entirely in the MINRES phase, where the cost per iteration is essentially the same as for MINRES. In the MINRES-QLP phase, one more work vector and $5n$ more multiplications are used per iteration.

MINRES-QLP described here is implemented in FORTRAN 90 for real double-precision problems. It contains no machine-dependent constants and does not need to use features such as polymorphism from FORTRAN 2003 or 2008. It requires an auxiliary subroutine `Aprod` and, if a preconditioner is supplied, a second subroutine `Msolve`. We also provide a complex implementation for Hermitian problems. Precision other than double can be obtained by changing one line of code. The programs can be compiled with FORTRAN 90 and FORTRAN 95 compilers such as `g95` and `gfortran`.

We also maintain a MATLAB implementation capable of solving both real and complex problems. All implementations are available at [SOL] or the first author's homepage, http://home.uchicago.edu/sctchoi/.

Table I lists the main notation used.

Table I. Key notation.

| | |
|---|---|
| $\Vert\cdot\Vert$ | matrix or vector two-norm |
| $\bar{A}$ | $\bar{A} = A - \sigma I$ (see also $\sigma$ below) |
| $\text{cond}(A)$ | condition number of $A$ with respect to two-norm $= \frac{\max\lvert\lambda_i\rvert}{\min_{\lambda_i \neq 0}\lvert\lambda_i\rvert}$ |
| $e_i$ | $i$th unit vector |
| $\ell$ | index of the last Lanczos iteration when $\beta_{\ell+1} = 0$ |
| $n$ | order of $A$ |
| $\text{null}(A)$ | null space of $A$ defined as $\{x \in \mathbb{R}^n \mid Ax = 0\}$ |
| $\text{range}(A)$ | column space of $A$ defined as $\{Ax \mid x \in \mathbb{R}^n\}$ |
| $^T$ | (right superscript to a vector or a matrix) transpose |
| $x^\dagger$ | unique minimum-length least-squares solution of problem (3) |
| $\mathcal{K}_k(A,b)$ | $k$th Krylov subspace defined as $\text{span}\{b, Ab, \ldots, A^{k-1}b\}$ |
| $\varepsilon$ | machine precision |
| $\sigma$ | real scalar shift to diagonal of $A$ |

## 1.1 Least-Squares Methods

Further existing methods that could be applied to (3) are CGLS and LSQR [Paige and Saunders 1982a; Paige and Saunders 1982b], LSMR [Fong and Saunders 2011], and GMRES [Saad and Schultz 1986], all of which reduce $\|r_k\|$ monotonically. The first three methods would require two products $Av_k$ and $Au_k$ each iteration and would be generating points in less favorable subspaces. GMRES requires only products $Av_k$ and could use any nonsingular (possibly indefinite) preconditioner. It

Table II. Comparison of various least-squares solvers on $n \times n$ systems (3). Storage refers to memory required by working vectors in the solvers. Work counts number of floating-point multiplications. On inconsistent systems, all solvers below except MINRES and GMRES with restart parameter $m$ return the minimum-length LS solution (assuming no preconditioner).

| Solver | Storage | Work per Iteration | Products per Iteration | Systems to Solve per Iteration with Preconditioner |
|---|---|---|---|---|
| MINRES | $7n$ | $9n$ | 1 | 1 |
| MINRES-QLP | $7n$–$8n$ | $9n$–$14n$ | 1 | 1 |
| GMRES($m$) | $(m+2)n$ | $(m+3+1/m)n$ | 1 | 1 |
| CGLS | $4n$ | $5n$ | 2 | 2 |
| LSQR | $5n$ | $8n$ | 2 | 2 |
| LSMR | $6n$ | $9n$ | 2 | 2 |

needs increasing storage and work each iteration, perhaps requiring restarts, but it could be more effective than MINRES or MINRES-QLP (and the other solvers) if few total iterations were required. Table II summarizes the computational requirements of each method.

### 1.2 Regularization

We do not discourage using CGLS, LSQR, or LSMR if the goal is to regularize an ill-posed problem using a small damping factor $\lambda > 0$ as follows:

$$\min_x \left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|. \tag{4}$$

However, this approach destroys the original problem's symmetry. The normal equation of (4) is $(A^2 + \lambda^2 I)x = Ab$, which suggests that a diagonal shift to $A$ may well serve the same purpose in some cases. For symmetric positive-definite $A$, $\bar{A} = A - \sigma I$ with $\sigma < 0$ enjoys a smaller condition number. When $A$ is indefinite, a good choice of $\sigma$ may not exist, for example, if the eigenvalues of $A$ were symmetrically positioned around zero. When this symmetric form is applicable, it is convenient in MINRES and MINRES-QLP; see (3), (5), and (15). We also remark that MINRES and MINRES-QLP produce good estimates of the largest and smallest singular values of $\bar{A}$ (via diagonal values of $R_k$ or $L_k$ in (7) and (11); see [Choi et al. 2011, Section 4]).

Three other regularization tools in the literature (see [Golub and Van Loan 1996, Sections 12.1.1-12.1.3] and [Hansen 1998]) are LSQI, cross-validation, and L-curve. LSQI involves solving a nonlinear equation and is not immediately compatible with the Lanczos framework. Cross-validation takes one row out at a time and thus does not preserve symmetry. The L-curve approach for a CG-type method takes iteration $k$ as the regularization parameter [Hansen 1998, Chapter 8] if both $\|r_k\|$ and $\|x_k\|$ are monotonic. By design, $\|r_k\|$ is monotonic in MINRES and MINRES-QLP, and so is $\|x_k\|$ when $\bar{A}$ is positive definite [Fong 2011]. Otherwise, we prefer the condition L-curve approach in [Calvetti et al. 2000], which graphs $\operatorname{cond}(T_k)$ against $\|r_k\|$. Yet another L-curve feasible in MINRES-QLP is $\|x^{(2)}_{k-2}\|$ against $\|r_k\|$, since the former is also monotonic (but available two iterations in lag); see Section 2.4.

## 2. MATHEMATICAL BACKGROUND

Notation and details of algorithmic development from [Choi 2006; Choi et al. 2011] are summarized here. As noted earlier, "$A$" in (1)–(3) is treated as $A - \sigma I$.

### 2.1 Lanczos Process

MINRES and MINRES-QLP use the symmetric Lanczos process [Lanczos 1950] to reduce $A$ to a tridiagonal form $\underline{T_k}$. The process is initialized with $v_0 \equiv 0$, $\beta_1 = \|b\|$, and $\beta_1 v_1 = b$. After $k$ steps of the tridiagonalization, we have produced

$$p_k = Av_k - \sigma v_k, \qquad \alpha_k = v_k^T p_k, \qquad \beta_{k+1} v_{k+1} = p_k - \alpha_k v_k - \beta_k v_{k-1}, \tag{5}$$

where we choose $\beta_k > 0$ to give $\|v_k\| = 1$. Numerically,

$$p_k = Av_k - \sigma v_k - \beta_k v_{k-1}, \qquad \alpha_k = v_k^T p_k, \qquad \beta_{k+1} v_{k+1} = p_k - \alpha_k v_k$$

is slightly better than (5) [Paige 1976], but we can express (5) in matrix form:

$$V_k \equiv \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}, \qquad AV_k = V_{k+1} \underline{T_k}, \qquad \underline{T_k} \equiv \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix}, \tag{6}$$

where $T_k = \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$, $i = 1, \dots, k$. In exact arithmetic, the Lanczos vectors in the columns of $V_k$ are orthonormal, and the process stops with $k = \ell$ when $\beta_{\ell+1} = 0$ for some $\ell \le n$, and then $AV_\ell = V_\ell T_\ell$. The rank of $T_\ell$ could be $\ell$ or $\ell - 1$ (see Theorem 2.2).

### 2.2 MINRES Phase

MINRES-QLP typically starts with a MINRES phase, which applies a series of reflectors $Q_k$ to transform $\underline{T_k}$ to an upper triangular matrix $\underline{R_k}$:

$$Q_k \begin{bmatrix} \underline{T_k} & \beta_1 e_1 \end{bmatrix} = \begin{bmatrix} R_k & t_k \\ 0 & \phi_k \end{bmatrix} \equiv \begin{bmatrix} \underline{R_k} & \bar{t}_{k+1} \end{bmatrix}, \tag{7}$$

where

$$Q_k = Q_{k,k+1} \begin{bmatrix} Q_{k-1} & \\ & 1 \end{bmatrix}, \qquad Q_{k,k+1} \equiv \begin{bmatrix} I_{k-1} & & \\ & c_k & s_k \\ & s_k & -c_k \end{bmatrix}.$$

In the $k$th step, $Q_{k,k+1}$ is effectively a Householder reflector of dimension 2 [Trefethen and Bau 1997, Exercise 10.4], and its action including its effect on later columns of $T_j$, $k < j \le \ell$, is compactly described by

$$\begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} \left[\begin{array}{ccc|c} \gamma_k & \delta_{k+1} & 0 & \phi_{k-1} \\ \beta_{k+1} & \alpha_{k+1} & \beta_{k+2} & 0 \end{array}\right] = \left[\begin{array}{ccc|c} \gamma_k^{(2)} & \delta_{k+1}^{(2)} & \epsilon_{k+2} & \tau_k \\ 0 & \gamma_{k+1} & \delta_{k+2} & \phi_k \end{array}\right],$$

where the superscripts with numbers in parentheses indicate the number of times the values have been modified. The $k$th solution approximation to (3) is then defined to be $x_k = V_k y_k$, where $y_k$ solves the subproblem

$$y_k = \arg\min_{y \in \mathbb{R}^k} \|\underline{T_k} y - \beta_1 e_1\| = \arg\min_{y \in \mathbb{R}^k} \|\underline{R_k} y - \bar{t}_{k+1}\|. \tag{8}$$

When $k < \ell$, $R_k$ is nonsingular and the unique solution of the above subproblem satisfies $R_k y_k = t_k$. Instead of solving for $y_k$, MINRES solves $R_k^T D_k^T = V_k^T$ by forward substitution, obtaining the last column $d_k$ of $D_k$ at iteration $k$. At the same time, it updates $x_k \in \mathcal{K}_k(A,b)$ (see Table I for definition) via $x_0 \equiv 0$ and

$$x_k = V_k y_k = D_k R_k y_k = D_k t_k = x_{k-1} + \tau_k d_k, \quad \tau_k \equiv e_k^T t_k, \tag{9}$$

where one can show using $V_k = D_k R_k$ that $d_k = (v_k - \delta_k^{(2)} d_{k-1} - \epsilon_k d_{k-2})/\gamma_k^{(2)}$.

## 2.3 MINRES-QLP Phase

The MINRES phase transfers to the MINRES-QLP phase when an estimate of the condition number of $A$ exceeds an input parameter *trancond*. Thus, $\mathit{trancond} > 1/\varepsilon$ leads to MINRES iterates throughout (where $\varepsilon \approx 10^{-16}$ denotes the floating-point precision), whereas $\mathit{trancond} = 1$ generates MINRES-QLP iterates from the start.

Suppose for now that there is no MINRES phase. Then MINRES-QLP applies left reflections as in (7) and a further series of right reflections to transform $R_k$ to a lower triangular matrix $L_k = R_k P_k$, where

$$P_k = P_{1,2} \quad P_{1,3}P_{2,3} \quad \cdots \quad P_{k-2,k}P_{k-1,k},$$

$$P_{k-2,k} = \begin{bmatrix} I_{k-3} & & & \\ & c_{k2} & & s_{k2} \\ & & 1 & \\ & s_{k2} & & -c_{k2} \end{bmatrix}, \qquad P_{k-1,k} = \begin{bmatrix} I_{k-2} & & \\ & c_{k3} & s_{k3} \\ & s_{k3} & -c_{k3} \end{bmatrix}.$$

In the $k$th step, the actions of $P_{k-2,k}$ and $P_{k-1,k}$ are compactly described by

$$\begin{aligned} &\begin{bmatrix} \gamma_{k-2}^{(5)} & & \epsilon_k \\ \vartheta_{k-1} & \gamma_{k-1}^{(4)} & \delta_k^{(2)} \\ & & \gamma_k^{(2)} \end{bmatrix} \begin{bmatrix} c_{k2} & & s_{k2} \\ & 1 & \\ s_{k2} & & -c_{k2} \end{bmatrix} \begin{bmatrix} 1 & & \\ & c_{k3} & s_{k3} \\ & s_{k3} & -c_{k3} \end{bmatrix} \\ = &\begin{bmatrix} \gamma_{k-2}^{(6)} & & \\ \vartheta_{k-1}^{(2)} & \gamma_{k-1}^{(4)} & \delta_k^{(3)} \\ \eta_k & & \gamma_k^{(3)} \end{bmatrix} \begin{bmatrix} 1 & & \\ & c_{k3} & s_{k3} \\ & s_{k3} & -c_{k3} \end{bmatrix} = \begin{bmatrix} \gamma_{k-2}^{(6)} & & \\ \vartheta_{k-1}^{(2)} & \gamma_{k-1}^{(5)} & \\ \eta_k & \vartheta_k & \gamma_k^{(4)} \end{bmatrix}. \end{aligned} \tag{10}$$

The $k$th approximate solution to (3) is then defined to be $x_k = V_k y_k = V_k P_k u_k = W_k u_k$, where $u_k$ solves the subproblem

$$u_k \equiv \arg\min_u \|u\| \quad \text{s.t.} \quad u \in \arg\min_{u \in \mathbb{R}^k} \left\| \begin{bmatrix} L_k \\ 0 \end{bmatrix} u - \begin{bmatrix} t_k \\ \phi_k \end{bmatrix} \right\|. \tag{11}$$

For $k < \ell$, $R_k$ and $L_k$ are nonsingular because $\underline{T_k}$ has full column rank by Lemma 2.1 below. It is only when $k = \ell$ and $b \notin \operatorname{range}(A)$ that $R_k$ and $L_k$ are singular with rank $\ell - 1$ by Theorem 2.2, in which case one can show that $\eta_k = \gamma_k^{(3)} = \vartheta_k = \gamma_k^{(4)} = 0$ in (10) and $L_\ell = \left[\begin{smallmatrix} L_{\ell-1} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ with $L_{\ell-1}$ nonsingular. In any case, we need to solve only the nonsingular lower triangular systems $L_k u_k = t_k$ or $L_{\ell-1} u_{\ell-1} = t_{\ell-1}$. Then, $u_k$ and $y_k = P_k u_k$ are the min-length solutions of (11) and (8), respectively.

MINRES-QLP updates $x_{k-2}$ to obtain $x_k$ by short-recurrence orthogonal steps:

$$x_{k-2}^{(2)} = x_{k-3}^{(2)} + \mu_{k-2}^{(3)} w_{k-2}^{(4)}, \text{ where } x_{k-3}^{(2)} \equiv W_{k-3}^{(4)} u_{k-3}^{(3)}, \tag{12}$$

$$x_k = x_{k-2}^{(2)} + \mu_{k-1}^{(2)} w_{k-1}^{(3)} + \mu_k w_k^{(2)}, \tag{13}$$

where $w_j$ refers to the $j$th column of $W_k = V_k P_k$ and $\mu_i$ is the $i$th element of $u_k$.

If this phase is preceded by a MINRES phase of $k$ iterations ($0 < k < \ell$), it starts by transferring the last three vectors $d_{k-2}$, $d_{k-1}$, $d_k$ to $w_{k-2}$, $w_{k-1}$, $w_k$, and the solution estimate $x_k$ from (9) to $x_{k-2}^{(2)}$ in (12). This needs the last two rows of $L_k u_k = t_k$ (to give $\mu_{k-1}$, $\mu_k$) and the relations $W_k = D_k L_k$ and $x_{k-2}^{(2)} = x_k - \mu_{k-1} w_{k-1} - \mu_k w_k$. The cheaply available right reflections $P_k$ and the bottom right $3 \times 3$ submatrix of $L_k$ (i.e., the last term in (10)) need to have been saved in the MINRES phase in order to facilitate the transfer.

### 2.4 Norm Estimates and Stopping Conditions

Short-term recurrences are used to estimate the following quantities (where we assume $\sigma = 0$ for simplicity):

$$\begin{aligned}
\|r_k\| &\approx \phi_k = \phi_{k-1} s_k, & \phi_0 &= \|b\| & &(\phi_k \searrow)\\
\|Ar_k\| &\approx \psi_k = \phi_k \|[\gamma_{k+1} \;\; \delta_{k+2}]\|, & & & &(\psi_\ell = 0)\\
\|x_k^{(2)}\| &\approx \chi_{k-2}^{(2)} = \|[\chi_{k-3}^{(2)} \;\; \mu_{k-2}^{(3)}]\|, & \chi_{-2} &= \chi_{-1} = 0 & &(\chi_{k-2}^{(2)} \nearrow)\\
\|x_k\| &\approx \chi_k = \|[\chi_{k-2}^{(2)} \;\; \mu_{k-1}^{(2)} \;\; \mu_k]\|, & \chi_0 &= 0 & &(\chi_\ell = \|x^\dagger\|)\\
\|Ax_k\| &\approx \omega_k = \|[\omega_{k-1} \;\; \tau_k]\|, & \omega_0 &= 0 & &(\omega_k \nearrow)\\
\|A\| &\approx \mathcal{A}_k = \max\left\{\mathcal{A}_{k-1}, \|\underline{T_k} e_k\|, \overline{\gamma}_k\right\}, & \mathcal{A}_0 &= 0 & &(\mathcal{A}_k \nearrow \|A\|)\\
\mathrm{cond}(A) &\approx \kappa_k = \mathcal{A}_k / \underline{\gamma}_k, & \kappa_0 &= 1 & &(\kappa_k \nearrow \mathrm{cond}(A))
\end{aligned}$$

where $\overline{\gamma}_k$ and $\underline{\gamma}_k$ are the largest and smallest diagonals of $L_k$ in absolute value. The up (down) arrows in parentheses indicate that the associated quantities increase (decrease) monotonically. The last two estimates tend to their targets from below; see [Choi 2006; Choi et al. 2011] for derivation.

MINRES-QLP has 14 possible stopping conditions in five classes that use the above estimates and optional input parameters *itnlim*, *rtol*, *Acondlim*, and *maxxnorm*:

(C1) From Lanczos and the QLP factorization:

$$k = \mathit{itnlim}; \qquad \beta_{k+1} < \varepsilon; \qquad \left|\gamma_k^{(4)}\right| < \varepsilon;$$

(C2) Normwise relative backward errors (NRBE) [Paige and Strakoš 2002]:

$$\|r_k\| / \left(\|A\|\|x_k\| + \|b\|\right) \le \max(\mathit{rtol}, \varepsilon); \qquad \|Ar_k\| / \left(\|A\|\|r_k\|\right) \le \max(\mathit{rtol}, \varepsilon);$$

(C3) Regularization attempts:

$$\mathrm{cond}(A) \ge \min(\mathit{Acondlim}, \mathit{0.1}/\varepsilon); \qquad \|x_k\| \ge \mathit{maxxnorm};$$

(C4) Degenerate cases:

$$\begin{aligned}
&\beta_1 = 0 \quad \Rightarrow \quad b \;= 0 \quad \Rightarrow \quad x = 0 \text{ is the solution;}\\
&\beta_2 = 0 \quad \Rightarrow \quad v_2 = 0 \quad \Rightarrow \quad Ab = \alpha_1 b,\\
&\qquad \text{i.e., } b \text{ and } \alpha_1 \text{ are an eigenpair of } A, \text{ and } x = b/\alpha_1 \text{ solves } Ax = b;
\end{aligned}$$

(C5) Erroneous inputs:

$$A \text{ not symmetric}; \qquad M \text{ not symmetric}; \qquad M \text{ not positive definite};$$

where $M$ is a preconditioner to be described in the next section. For symmetry of $A$, it is not practical to check $e_i^T A e_j = e_j^T A e_i$ for all $i, j = 1, \dots, n$. Instead, we statistically test whether $z = |x^T(Ay) - y^T(Ax)|$ is sufficiently small for two nonzero $n$-vectors $x$ and $y$ (e.g., each element in the vectors is drawn from the standard normal distribution). For positive definiteness of $M$, since $M$ is positive definite if and only if $M^{-1}$ is positive definite, we simply test that $z_k^T M^{-1} z_k = z_k^T q_k > 0$ each iteration (see Section 3).

We find that the recurrence relations for $\phi_k$ and $\psi_k$ hold to high accuracy. Thus $x_k$ is an acceptable solution of (3) if the computed value of $\phi_k$ or $\psi_k$ is suitably small according to the NRBE tests in class (C2) above. When a condition in (C3) is met, the final $x_k$ may or may not be an acceptable solution.

The class (C1) tests for small $\beta_{k+1}$ and $\gamma_k^{(4)}$ are included in the unlikely case in practice that the theoretical Lanczos termination occurs. Ideally one of the NRBE tests should cause MINRES-QLP to terminate. If not, it is an indication that the problem is very ill-conditioned, in which case the regularization and preconditioning techniques of Sections 1.2 and 3 may be helpful.

### 2.5 Two Theorems

We complete this section by presenting two theorems from [Choi et al. 2011] with slightly simpler proofs.

LEMMA 2.1. rank$(\underline{T_k}) = k$ *for all* $1 \le k < \ell$.

PROOF. For $1 \le k < \ell$ we have $\beta_2, \dots, \beta_{k+1} > 0$ by definition. Hence $\underline{T_k}$ has full column rank. □

THEOREM 2.2. *$T_\ell$ is nonsingular if and only if $b \in$ range$(A)$. Furthermore,* rank$(T_\ell) = \ell - 1$ *if* $b \notin$ range$(A)$.

PROOF. We use $AV_\ell = V_\ell T_\ell$ twice. First, if $T_\ell$ is nonsingular, we can solve $T_\ell y_\ell = \beta_1 e_1$ and then $AV_\ell y_\ell = V_\ell T_\ell y_\ell = V_\ell \beta_1 e_1 = b$. Conversely, if $b \in \text{range}(A)$, then $\text{range}(V_\ell) \subseteq \text{range}(A)$. Suppose $T_\ell$ is singular. Then there exists $z \neq 0$ such that $V_\ell T_\ell z = AV_\ell z = 0$. That is, $0 \neq V_\ell z \in \text{null}(A)$. But this is impossible because $V_\ell z \in \text{range}(A)$ and $\text{null}(A) \cap \text{range}(V_\ell) = 0$. Thus, $T_\ell$ must be nonsingular.

We have shown that if $b \notin \text{range}(A)$, $T_\ell = \begin{bmatrix} \underline{T_{\ell-1}} & \begin{smallmatrix} \beta_\ell e_{\ell-1} \\ \alpha_\ell \end{smallmatrix} \end{bmatrix}$ is singular, and therefore $\ell > \text{rank}(T_\ell) \ge \text{rank}(\underline{T_{\ell-1}}) = \ell - 1$ by Lemma 2.1. Therefore, $\text{rank}(T_\ell) = \ell - 1$. □

By Lemma 2.1 and Theorem 2.2 we are assured that the QLP decomposition without column pivoting [Stewart 1999; Choi et al. 2011] for $\underline{T_k}$ is rank-revealing, which is a necessary precondition for solving a least-squares problem.

THEOREM 2.3. *In MINRES-QLP, $x_\ell$ is the minimum-length solution of (3).*

PROOF. $y_\ell$ comes from the min-length LS solution of $T_\ell y_\ell \approx \beta_1 e_1$ and thus satisfies the normal equation $T_\ell^2 y_\ell = T_\ell \beta_1 e_1$ and $y_\ell \in \text{range}(T_\ell)$. Now $x_\ell = V_\ell y_\ell$ and $Ax_\ell = AV_\ell y_\ell = V_\ell T_\ell y_\ell$. Hence $A^2 x_\ell = AV_\ell T_\ell y_\ell = V_\ell T_\ell^2 y_\ell = V_\ell T_\ell \beta_1 e_1 = Ab$. Thus $x_\ell$ is an LS solution of (3). Since $y_\ell \in \text{range}(T_\ell)$, $y_\ell = T_\ell z$ for some $z$, and so $x_\ell = V_\ell y_\ell = V_\ell T_\ell z = AV_\ell z \in \text{range}(A)$ is the min-length LS solution of (3). □

## 3. PRECONDITIONING

Iterative methods can be accelerated if preconditioners are available and well-chosen. For MINRES-QLP, we want to choose a symmetric positive-definite matrix $M$ to solve a nonsingular system (1) by implicitly solving an equivalent symmetric consistent system $M^{-\frac{1}{2}}AM^{-\frac{1}{2}}\bar{x} = \bar{b}$, where $M^{\frac{1}{2}}x = \bar{x}$, $\bar{b} = M^{-\frac{1}{2}}b$, and $\mathrm{cond}(M^{-\frac{1}{2}}AM^{-\frac{1}{2}}) \ll \mathrm{cond}(A)$. This two-sided preconditioning preserves symmetry. Thus we can derive preconditioned MINRES-QLP by applying MINRES-QLP to the equivalent problem and setting $x = M^{-\frac{1}{2}}\bar{x}$.

With preconditioned MINRES-QLP, we can solve a singular consistent system (2), but we will obtain a least-squares solution that is not necessarily the minimum-length solution (unless $M = I$). For inconsistent systems (3), preconditioning alters the least-squares norm to $\|\cdot\|_{M^{-1}}$, and the solution is of minimum length in the new norm space. We refer readers to [Choi et al. 2011, Section 7] for a detailed discussion of various approaches to preserving the two-norm "minimum length."

To derive MINRES-QLP, we define

$$z_k = \beta_k M^{\frac{1}{2}} v_k, \qquad q_k = \beta_k M^{-\frac{1}{2}} v_k, \qquad \text{so that} \quad M q_k = z_k. \tag{14}$$

Then $\beta_k = \|\beta_k v_k\| = \|M^{-\frac{1}{2}} z_k\| = \|z_k\|_{M^{-1}} = \|q_k\|_M = \sqrt{q_k^T z_k}$, where the square root is well defined because $M$ is positive definite, and the following expressions replace the quantities in (5) in the Lanczos iterations:

$$p_k = Aq_k - \sigma q_k, \qquad \alpha_k = \frac{1}{\beta_k^2} q_k^T p_k, \qquad z_{k+1} = \frac{1}{\beta_k} p_k - \frac{\alpha_k}{\beta_k} z_k - \frac{\beta_k}{\beta_{k-1}} z_{k-1}. \tag{15}$$

We also need to solve the system $Mq_k = z_k$ in (14) at each iteration.

In the MINRES phase, we define $\bar{d}_k = M^{-\frac{1}{2}} d_k$ and update the solution of the original problem (1) by

$$\bar{d}_k = \Big(\frac{1}{\beta_k} q_k - \delta_k^{(2)} \bar{d}_{k-1} - \epsilon_k \bar{d}_{k-2}\Big) \big/ \gamma_k^{(2)}, \qquad x_k = M^{-\frac{1}{2}} \bar{x}_k = x_{k-1} + \tau_k \bar{d}_k.$$

In the MINRES-QLP phase, we define $\overline{W}_k \equiv M^{-\frac{1}{2}} W_k = (M^{-\frac{1}{2}} V_k) P_k$ and update the solution estimate of problem (1) by orthogonal steps:

$$\begin{aligned}
\bar{w}_k &= -(c_{k2}/\beta_k) q_k + s_{k2} \bar{w}_{k-2}^{(3)}, & \bar{w}_{k-2}^{(4)} &= (s_{k2}/\beta_k) q_k + c_{k2} \bar{w}_{k-2}^{(3)}, \\
\bar{w}_k^{(2)} &= s_{k3} \bar{w}_{k-1}^{(2)} - c_{k3} \bar{w}_k, & \bar{w}_{k-1}^{(3)} &= c_{k3} \bar{w}_{k-1}^{(2)} + s_{k3} \bar{w}_k, \\
x_{k-2}^{(2)} &= x_{k-3}^{(2)} + \mu_{k-2}^{(3)} \bar{w}_{k-2}^{(4)}, & x_k &= x_{k-2}^{(2)} + \mu_{k-1}^{(2)} \bar{w}_{k-1}^{(3)} + \mu_k \bar{w}_k^{(2)}.
\end{aligned}$$

Let $\bar{r}_k = \bar{b} - M^{-\frac{1}{2}}(A - \sigma I) M^{-\frac{1}{2}} \bar{x}_k = M^{-\frac{1}{2}} r_k$. Then $x_k = M^{-\frac{1}{2}} \bar{x}_k$ is an acceptable solution of (1) if the computed value of $\phi_k \approx \|\bar{r}_k\| = \|r_k\|_{M^{-1}}$ is sufficiently small.

We can now present our pseudocode in Algorithm 1. The reflectors are implemented in Algorithm 2 SymOrtho$(a, b)$ for real $a$ and $b$, which is a stable form for computing $r = \sqrt{a^2 + b^2} \geq 0$ , $c = \frac{a}{r}$, and $s = \frac{b}{r}$. The complexity is at most 6 flops and a square root. Algorithm 1 lists all steps of MINRES-QLP with preconditioning. For simplicity, $\bar{w}_k$ is written as $w_k$ for all relevant $k$. Also, the output $x$ solves $(A - \sigma I)x \approx b$, but other outputs are associated with the preconditioned system.

**Algorithm 1:** Pseudocode of preconditioned MINRES-QLP for solving $(A-\sigma I)x \approx b$. In the right-justified comments, $\tilde{A} \equiv M^{-\frac{1}{2}}(A-\sigma I)M^{-\frac{1}{2}}$.

**input**: $A, b, \sigma, M$

$z_0 = 0, \quad z_1 = b, \quad$ Solve $Mq_1 = z_1, \quad \beta_1 = \sqrt{b^T q_1}, \quad \phi_0 = \beta_1$ `[Initialize]`

$w_0 = w_{-1} = 0, \qquad x_{-2} = x_{-1} = x_0 = 0$

$c_{0,1} = c_{0,2} = c_{0,3} = -1, \quad s_{0,1} = s_{0,2} = s_{0,3} = 0, \quad \tau_0 = \omega_0 = \chi_{-2} = \chi_{-1} = \chi_0 = 0$

$\kappa_0 = 1, \quad \mathcal{A}_0 = \delta_1 = \gamma_{-1} = \gamma_0 = \eta_{-1} = \eta_0 = \eta_1 = \vartheta_{-1} = \vartheta_0 = \vartheta_1 = \mu_{-1} = \mu_0 = 0$

$k = 0$

**while** no stopping condition is satisfied **do**

- $k \leftarrow k+1$
- $p_k = Aq_k - \sigma q_k, \qquad \alpha_k = \frac{1}{\beta_k^2} q_k^T p_k$ `[Preconditioned Lanczos]`
- $z_{k+1} = \frac{1}{\beta_k} p_k - \frac{\alpha_k}{\beta_k} z_k - \frac{\beta_k}{\beta_{k-1}} z_{k-1}$
- Solve $Mq_{k+1} = z_{k+1}, \qquad \beta_{k+1} = \sqrt{q_{k+1}^T z_{k+1}}$
- **if** $k=1$ **then** $\rho_k = \|[\alpha_k \;\; \beta_{k+1}]\|$ **else** $\rho_k = \|[\beta_k \;\; \alpha_k \;\; \beta_{k+1}]\|$
- $\delta_k^{(2)} = c_{k-1,1}\delta_k + s_{k-1,1}\alpha_k$ `[Previous left reflection...]`
- $\gamma_k = s_{k-1,1}\delta_k - c_{k-1,1}\alpha_k$ `[on middle two entries of` $\underline{T_k} e_k$ `...]`
- $\epsilon_{k+1} = s_{k-1,1}\beta_{k+1}$ `[produces first two entries in` $\underline{T_{k+1}} e_{k+1}$`]`
- $\delta_{k+1} = -c_{k-1,1}\beta_{k+1}$
- $c_{k1}, s_{k1}, \gamma_k^{(2)} \leftarrow \mathrm{SymOrtho}(\gamma_k, \beta_{k+1})$ `[Current left reflection]`
- $c_{k2}, s_{k2}, \gamma_{k-2}^{(6)} \leftarrow \mathrm{SymOrtho}(\gamma_{k-2}^{(5)}, \epsilon_k)$ `[First right reflection]`
- $\delta_k^{(3)} = s_{k2}\vartheta_{k-1} - c_{k2}\delta_k^{(2)}, \qquad \gamma_k^{(3)} = -c_{k2}\gamma_k^{(2)}, \qquad \eta_k = s_{k2}\gamma_k^{(2)}$
- $\vartheta_{k-1}^{(2)} = c_{k2}\vartheta_{k-1} + s_{k2}\delta_k^{(2)}$
- $c_{k3}, s_{k3}, \gamma_{k-1}^{(5)} \leftarrow \mathrm{SymOrtho}(\gamma_{k-1}^{(4)}, \delta_k^{(3)})$ `[Second right reflection...]`
- $\vartheta_k = s_{k3}\gamma_k^{(3)}, \qquad \gamma_k^{(4)} = -c_{k3}\gamma_k^{(3)}$ `[to zero out` $\delta_k^{(3)}$`]`
- $\tau_k = c_{k1}\phi_{k-1}$ `[Last element of` $t_k$`]`
- $\phi_k = s_{k1}\phi_{k-1}, \quad \psi_{k-1} = \phi_{k-1}\|[\gamma_k \;\; \delta_{k+1}]\|$ `[Update` $\|\bar{r}_k\|$, $\|\tilde{A}\bar{r}_{k-1}\|$`]`
- **if** $k=1$ **then** $\gamma_{\min} = \gamma_1$ **else** $\gamma_{\min} \leftarrow \min\{\gamma_{\min}, \gamma_{k-2}^{(6)}, \gamma_{k-1}^{(5)}, |\gamma_k^{(4)}|\}$
- $\mathcal{A}_k = \max\{\mathcal{A}_{k-1}, \rho_k, \gamma_{k-2}^{(6)}, \gamma_{k-1}^{(5)}, |\gamma_k^{(4)}|\}$ `[Update` $\|\tilde{A}\|$`]`
- $\omega_k = \|[\omega_{k-1} \;\; \tau_k]\|, \qquad \kappa_k \leftarrow \mathcal{A}_k/\gamma_{\min}$ `[Update` $\|\tilde{A}x_k\|$, $\mathrm{cond}(\tilde{A})$`]`
- $w_k = -(c_{k2}/\beta_k)q_k + s_{k2}w_{k-2}^{(3)}$ `[Update` $w_{k-2}$, $w_{k-1}$, $w_k$`]`
- $w_{k-2}^{(4)} = (s_{k2}/\beta_k)q_k + c_{k2}w_{k-2}^{(3)}$
- **if** $k>2$ **then** $w_k^{(2)} = s_{k3}w_{k-1}^{(2)} - c_{k3}w_k, \qquad w_{k-1}^{(3)} = c_{k3}w_{k-1}^{(2)} + s_{k3}w_k$
- **if** $k>2$ **then** $\mu_{k-2}^{(3)} = (\tau_{k-2} - \eta_{k-2}\mu_{k-4}^{(4)} - \vartheta_{k-2}\mu_{k-3}^{(3)})/\gamma_{k-2}^{(6)}$ `[Update` $\mu_{k-2}$`]`
- **if** $k>1$ **then** $\mu_{k-1}^{(2)} = (\tau_{k-1} - \eta_{k-1}\mu_{k-3}^{(3)} - \vartheta_{k-1}^{(2)}\mu_{k-2}^{(3)})/\gamma_{k-1}^{(5)}$ `[Update` $\mu_{k-1}$`]`
- **if** $\gamma_k^{(4)} \neq 0$ **then** $\mu_k = (\tau_k - \eta_k\mu_{k-2}^{(3)} - \vartheta_k\mu_{k-1}^{(2)})/\gamma_k^{(4)}$ **else** $\mu_k = 0$ `[Compute` $\mu_k$`]`
- $x_{k-2}^{(2)} = x_{k-3}^{(2)} + \mu_{k-2}^{(3)}w_{k-2}^{(3)}$ `[Update` $x_{k-2}$`]`
- $x_k = x_{k-2}^{(2)} + \mu_{k-1}^{(2)}w_{k-1}^{(3)} + \mu_k w_k^{(2)}$ `[Compute` $x_k$`]`
- $\chi_{k-2}^{(2)} = \|[\chi_{k-3}^{(2)} \;\; \mu_{k-2}^{(3)}]\|$ `[Update` $\|x_{k-2}\|$`]`
- $\chi_k = \|[\chi_{k-2}^{(2)} \;\; \mu_{k-1}^{(2)} \;\; \mu_k]\|$ `[Compute` $\|x_k\|$`]`

$x = x_k, \quad \phi = \phi_k, \quad \psi = \phi_k\|[\gamma_{k+1} \;\; \delta_{k+2}]\|, \quad \chi = \chi_k, \quad \mathcal{A} = \mathcal{A}_k, \quad \omega = \omega_k$

**output**: $x, \phi, \psi, \chi, \mathcal{A}, \kappa, \omega$

**Algorithm 2:** Algorithm SymOrtho.

**input**: $a, b$

**if** $b = 0$ **then** $s = 0, \quad r = |a|$
  **if** $a = 0$ **then** $c = 1$ **else** $c = \text{sign}(a)$
**else if** $a = 0$ **then**
  $c = 0, \quad s = \text{sign}(b), \quad r = |b|$
**else if** $|b| \geq |a|$ **then**
  $\tau = a/b, \quad s = \text{sign}(b)/\sqrt{1+\tau^2}, \quad c = s\tau, \quad r = b/s$
**else if** $|a| > |b|$ **then**
  $\tau = b/a, \quad c = \text{sign}(a)/\sqrt{1+\tau^2}, \quad s = c\tau, \quad r = a/c$

**output**: $c, s, r$

## 4. KEY FORTRAN 90 DESIGN FEATURES

In this section we describe the key features of our FORTRAN implementations. For an accessible reference on the language syntax in various FORTRAN releases, we refer readers to [Chivers and Sleightholme 2006].

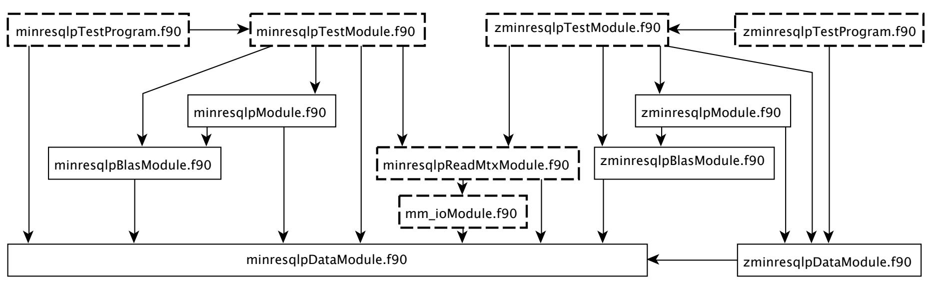


Fig. 1. FORTRAN 90 source files and their dependencies. Filenames boxed in broken lines are optional, and the corresponding files are used mainly for testing and demonstration.

Our FORTRAN 90 package contains the following files for symmetric problems with the first three files forming the core. Their dependencies are depicted in Figure 1.

1. `minresqlpDataModule.f90`: defines integer and real precision and constants used in other modules
2. `minresqlpBlasModule.f90`: packages FORTRAN 90 versions of some BLAS functions [BLAS]
3. `minresqlpModule.f90`: implements MINRES-QLP with preconditioning
4. `mm_ioModule.f90` and `minresqlpReadMtxModule.f90`: packages subroutines for reading Matrix Market files [Matrix Market] adapted from [Burkardt]
5. `minresqlpTestModule.f90`: illustrates how MINRES-QLP can call `Aprod` or `Msolve` with a short fixed parameter list, even if it needs arbitrary other data
6. `minresqlpTestProgram.f90`: contains the main driver program for unit tests
7. `Makefile`: compiles the FORTRAN source files via the Unix command `make`

Listing 1. Partial FORTRAN 90 code listing of minresqlpDataModule.

```
module minresqlpDataModule
 implicit none

 intrinsic                              :: selected_real_kind

 integer,          parameter, public :: dp = selected_real_kind(15)

 real(kind=dp), parameter, public :: zero = 0.0_dp, one = 1.0_dp

end module minresqlpDataModule
```

8. `readme.txt`: contains information about software license, other files in the package, and program compilation and execution.

The counterparts of these programs for Hermitian problems have the same filenames prefixed with the letter "`z`".

We review and step through the code in the following subsections. The line numbers in Listings 1–3 are used for reference only and do not correspond to actual line numbers in the source code. The vertical dots in Listing 2 lines 35 and 43 indicate omitted code of one or more lines. We also note that FORTRAN 90 keywords are displayed in bold in the listings, and that comments are marked with exclamation marks in italics.

### 4.1 Overloaded Intrinsic Operators and BLAS Procedures

For standard vector operations, we simply apply the intrinsic arithmetic and assignment operators $\pm, \times, =$. In addition we adopt a FORTRAN 90 translation [Burkardt] of two external level-1 BLAS functions `ddot` and `dnrm2` [BLAS] for computing inner products and two norms of vectors, which take care to avoid undesirable overflow or underflow.

In `minresqlpModule`, the line "`use minresqlpBlasModule`" can be omitted if the code is already linked to a BLAS library.

### 4.2 Using Modules and Interface and Passing User-Defined Subroutines to `MINRESQLP`

In our FORTRAN 90 implementation, we use *modules* instead of the obsolete FORTRAN 77 COMMON blocks for grouping programs units and data together and controlling their availability to other program units. A module can use `public` data and subroutines from other modules (by declaring an *interface* block), share its own `public` data and subroutines with other program units, and hide its own `private` data and subroutines from being used by other program units. We can also use modules to package procedures.

In Listing 2, line 2, module `minresqlpModule` uses the external `public` constant `dp` from `minresqlpDataModule`. From line 9 onwards, `minresqlpModule` defines a `public` subroutine `MINRESQLP`, where we implement MINRES-QLP in Algorithm 1.

A FORTRAN subroutine may have multiple and optional input and output arguments, which transfer information to and from a calling program. `MINRESQLP` has

Listing 2. Partial code listing of subroutine MINRESQLP in minresqlpModule.

```
module minresqlpModule
 use   minresqlpDataModule, only : dp, one, zero
 use   minresqlpBlasModule, only : dnrm2, ddot

 implicit none

 public    :: MINRESQLP, SYMORTHO
contains
  subroutine MINRESQLP(                                              &
            n, Aprod, b, shift, Msolve, disable, nout,               &
            itnlim, rtol, maxxnorm, trancond, Acondlim,              &
            x, istop, itn, rnorm, Arnorm, xnorm, Anorm, Acond )

    ! Inputs
    integer(ip), intent(in)                  :: n
    real(dp),    intent(in)                  :: b(n)
    integer(ip), intent(in), optional        :: itnlim, nout
    logical,     intent(in), optional        :: disable
    real(dp),    intent(in), optional        :: shift
    real(dp),    intent(in), optional        :: rtol, maxxnorm,
        trancond, Acondlim

   ! Outputs
    real(dp),    intent(out)                 :: x(n)
    integer(ip), intent(out), optional       :: istop, itn
    real(dp),    intent(out), optional       :: rnorm, Arnorm, xnorm,
        Anorm, Acond

    interface
     subroutine Aprod (n,x,y)                           ! y := Ax
       use       minresqlpDataModule
       integer,  intent(in)    :: n
       real(dp), intent(in)    :: x(n)
       real(dp), intent(out)   :: y(n)
     end subroutine Aprod
    ⋮
    end interface

    intrinsic :: abs, epsilon, sqrt

    ! Local arrays and variables
    real(dp)  :: r1(n), r2(n), v(n), w(n), wl(n),                    &

    ⋮
 end subroutine MINRESQLP
 ⋮
end module minresqlpModule
```

Argonne National Laboratory Preprint ANL/MCS-P3027-0812.

Listing 3. Partial code listing of minresqlpTestModule.

```
module minresqlpTestModule
  use  minresqlpDataModule, only : dp
  use  minresqlpModule,     only : MINRESQLP

  implicit none

  public   :: minresqlptest
  private  :: Aprod, Msolve

  ! DYNAMIC WORKSPACE DEFINED HERE.
  ! It is allocated in minresqlptest and used by Aprod or Msolve.

  real(dp), allocatable :: d(:)    !Defines diagonal matrix D.
  real(dp)              :: Ashift  !Shift diagonal elements of D in Msolve.
  real(dp)              :: Mpert   !Perturbation to D in Msolve
                                   !to avoid having an exact preconditioner.
contains
  subroutine Aprod(n,x,y)

    integer,  intent(in)    :: n
    real(dp), intent(in)    :: x(n)
    real(dp), intent(out)   :: y(n)

    integer  :: i

    do i = 1, n
       y(i) = d(i)*x(i)
    end do

  end subroutine Aprod

 ⋮

  subroutine minresqlptest( n, precon, shift, pertM, nout )
  ⋮
   call MINRESQLP(n, Aprod, b, shift, Msolve, disable,      &
         nout, itnlim, rtol, maxxnorm, trancond, Acondlim, &
         x, istop, itn, rnorm, Arnorm, xnorm, Anorm, Acond )

  ⋮
  end subroutine minresqlptest
end module minresqlpTestModule
```

a total of 20 arguments (see lines 9-12). The data types and `intent` of these arguments are declared in lines 15-25. For example, the first argument `n` in line 15 is an input integer, whereas `x(n)` in line 23 is an output $n$-vector of double precision.

Two input arguments `Aprod` and `Msolve` are external user-defined subroutines (Listing 3, lines 8 and 18-32) being passed into `MINRESQLP` as inputs—we recommend they be `private` for data integrity. The subroutine `Aprod` defines the matrix $A$ as an operator (in Algorithm 1, line 8). For a given vector $x$, the FORTRAN statement `call Aprod(n, x, y)` must return the product $y = Ax$ without altering the vector $x$. The subroutine `Msolve` is optional, and it defines a symmetric positive-definite matrix as an operator $M$ that serves as a preconditioner (line 10 in Algorithm 1). For a given vector $y$, the FORTRAN statement `call MSolve(n, y, x)` must solve the linear system $Mx = y$ without altering the vector $y$. To provide the compiler the necessary information about these `private` subroutines defined in `minresqlpTestModule`, an `interface` block in subroutine `MINRESQLP` is declared (lines 28-36 in Listing 2), which essentially replicates the headers of `Aprod` and `Msolve` in `minresqlpTestModule` (lines 18-32 in Listing 3).

`MINRESQLP` is called by the public routine `minresqlptest` defined in module `minresqlpTestModule` (see lines 7, 34-41 in Listing 3). Since `MINRESQLP` is public (Listing 2, line 7), `minresqlpTestModule` can simply `use` it (Listing 3, line 3). We have not listed details of `minresqlptest`, but it calls `MINRESQLP` with `Aprod` and `Msolve` passed as parameters (Listing 3, line 36).

We note that subroutine arrays and variables such as `r1(n)` in Listing 2, line 41, and `i` in Listing 3, line 24, are by default private and not accessible to other program units. In contrast, module arrays and variables are by default public and accessible to other program units. We have marked `d(:), Ashift, Mpert` as `private` in Listing 3, lines 13–15, in order to make them accessible to only the subroutines `minresqlptest`, `Aprod`, and `Msolve` in the containing module but not outside.

To summarize, we have described and provided a pattern that allows MINRES-QLP users to solve different problems by simply editing `minresTestModule` (and possibly the main program `minresTestProgram`, which calls `minresqlptest`). Users do not need to change `MINRESQLP` as long as the header of subroutines `Aprod` and `Msolve` stay the same in `minresTestModule`. If necessary, local arrays or variables such as `d(:)` can be used instead of additional input arguments to define these operators. In this way, users can make the data $A$ and $M$ known to `MINRESQLP` but hidden and thus secure from other programs.

Our design spares users from implementing *reverse communication*, in which the solver would return control to the calling program whenever `Aprod` or `Msolve` were to be invoked. (While reverse communication is widely used in scientific computing with FORTRAN 77, the resulting code usually appears formidable and unrecognizable from the original pseudocode; see [Dongarra et al. 1995] and [Oliveira and Stewart 2006] for two examples of CG and numerical integration coded in FORTRAN 77 and 90, respectively.) Our MINRES-QLP implementation achieves the purpose of reverse communication while preserving code readability and thus maintainability. The FORTRAN 90 module structure allows a user's $Ax$ products and $Mx = y$ solves to be implemented outside MINRES-QLP in the same way that MATLAB's function handles operate.

### 4.3 Unit Testing

Unit testing is an important software development strategy that cannot be overemphasized, especially in the scientific computing communities. Unit testing usually consists of multiple small and fast but specific and illuminating test cases that check whether the code behaves as designed. Software development is incremental, and errors (also known as bugs) are often found over time. Adding new functionalities or fixing errors often breaks the code for some earlier successful test cases. It is therefore critical to expand the test cases and to ensure that all unit tests are executed with expected results every time a key program unit is updated.

In our development of FORTRAN 90 MINRES-QLP, we have created a suite of 117 test cases including singular matrices representative of real-world applications [Foster 2009; Davis and Hu 2011]. The test program outputs results to `MINRESQLP.txt`. If users need to modify subroutine `MINRESQLP`, they can run these test cases and search for the word "appear" in the output file to check whether all tests are reported to be successful. For more sophisticated unit testing frameworks employed in large-scale scientific software development, see [O'Boyle et al. 2008].

### 4.4 Miscellaneous Issues

The complex program units for Hermitian linear systems and LS problems are similar to the real ones, and thus we will not go into detail. Many variables of type `real(dp)` are changed to `complex(dp)`.

To use a different precision throughout the program units, MINRES-QLP users can simply edit the input argument value of `dp` in `minresqlpDataModule`, line 6.

In the main subroutine `MINRESQLP`, we provide a logical parameter `debug` as a diagnostic tool; when it is true, variable values are printed to the standard output.

## 5. INPUTS, OUTPUTS, AND NUMERICAL EXAMPLES

Subroutine `MINRESQLP` contains the core implementation of MINRES-QLP and has 12 input parameters documented in the code as well as in Table III. It uses seven local $n$-vectors and returns a computed solution `x` as one of the eight outputs. Mandatory inputs are $n$, `Aprod`, and $b$. All outputs other than $x$ are optional. If an input is optional, MINRES-QLP prescribes a default value. It is well known that careful choice of parameter values is critical in the convergence behavior of iterative solvers. While the default parameter values in MINRES-QLP work well in most tests, they may need to be fine-tuned by trial and error, and for some applications it may be worthwhile to implement full or partial reorthogonalization of the Lanczos vectors [Simon 1984].

Table III: Input parameters in subroutine MINRES-QLP.

| **Input** | **Definition** |
| --- | --- |
| $n$ | The dimension of the symmetric matrix or operator $A$. |

Table III: Input parameters in MINRES-QLP (continued).

| **Input** | **Definition** |
|---|---|
| $b(n)$ | The right-hand-side vector $b$. |
| `Aprod` | An external subroutine defining the matrix $A$. For a given vector $x$, the statement `call Aprod ( n, x, y )` must return the product $y = Ax$ without altering the vector $x$. An extra call of `Aprod` is used to check if $A$ is symmetric. The program calling MINRES-QLP must declare `Aprod` to be external. |
| `Msolve` | An optional external subroutine defining a preconditioner $M$, which should approximate $A - \mathit{shift}I$ in some sense. If present, $M$ must be symmetric positive definite. For a given vector $x$, the statement `call Msolve( n, x, y )` must solve the linear system $My = x$ without altering the vector $x$. In general, $M$ should be chosen so that $\tilde{A} \equiv M^{-\frac{1}{2}}\bar{A}M^{-\frac{1}{2}}$ has more clustered eigenvalues. If $\bar{A}$ is positive definite, $\tilde{A}$ would ideally be close to a multiple of $I$. If $\bar{A}$ is indefinite, $\tilde{A}$ might be close to a multiple of $\mathtt{diag}(I \quad -I)$. If $M$ is absent, no preconditioner is applied. |
| *shift* | Should be zero if the system $Ax = b$ is to be solved. Otherwise, it could be an approximation to an eigenvalue of $A$, such as the Rayleigh quotient $(b^TAb)/(b^Tb)$ corresponding to the vector $b$. If $b$ is sufficiently like an eigenvector corresponding to an eigenvalue near *shift*, then the computed $x$ may have very large components. When normalized, $x$ may be closer to an eigenvector than $b$. Default value is 0. |
| *nout* | A file number. The calling program must open a file for output using for example: `open(nout, file='MINRESQLP.txt', status='unknown')`. If $\mathit{nout} > 0$, a summary of the iterations will be printed on unit *nout*. If *nout* is absent or the file associated with *nout* is not opened properly, results will be written to `MINRESQLP_tmp.txt`. Default value is 10. |
| *itnlim* | An upper limit on the number of iterations. Default to $4n$. |
| *rtol* | A user-specified tolerance. MINRES-QLP terminates if it appears that $\|\bar{r}\|$ is smaller than $\mathit{rtol}(\|\bar{A}\|\|\bar{x}\| + \|b\|)$, where $\bar{r} = \bar{b} - \bar{A}\bar{x}$, or that $\|\bar{A}\bar{r}\|$ is smaller than $\mathit{rtol}\|\bar{A}\|\|\bar{r}\|$. If $\mathit{shift} = 0$ and `Msolve` is absent, MINRES-QLP terminates if $\|r\|$ is smaller than $\mathit{rtol}(\|A\|\|x\| + \|b\|)$, where $r = b - Ax$, or if $\|Ar\|$ is smaller than $\mathit{rtol}\|A\|\|r\|$. Default to $\varepsilon$. |

Table III: Input parameters in MINRES-QLP (continued).

| **Input** | **Definition** |
|---|---|
| *maxxnorm* | An upper bound on $\|x\|$. Default value is $10^7$. |
| *Acondlim* | An upper bound on *Acond*, an estimate of cond$(A)$. Default value is $10^{15}$. |
| *trancond* | If *trancond* $> 1$, a switch is made from MINRES iterations to MINRES-QLP iterations when *Acond* $\geq$ *trancond*. If *trancond* $= 1$, all iterations will be MINRES-QLP iterations. If *trancond* $=$ *acondlim*, all iterations will be conventional MINRES iterations (which are slightly cheaper). Default value is $10^7$. |

We use two small examples to illustrate the output of `MINRESQLP`. [Choi 2006, Chapter 4] or [Choi et al. 2011, Section 8] give more significant numerical examples.

Table IV compares the MINRES solution to the MINRES-QLP solution for the small problem $Ax \approx b$, where $A = \text{diag}\left([1, \ldots, 10, 0]\right)$ and $b$ is a vector of all ones. Clearly, all but the last components are the same (in general, all components are different), and MINRES-QLP gives the minimum-length solution, whereas MINRES returns a minimum residual solution.

Table IV: MINRES and MINRES-QLP solutions of $Ax \approx e$, where $A = \text{diag}[1, \ldots, 10, 0]$.

| **MINRES** | **MINRES-QLP** |
|---|---|
| 1.000000000000001 | 1.000000000000000 |
| 0.500000000000001 | 0.500000000000001 |
| 0.333333333333333 | 0.333333333333333 |
| 0.250000000000001 | 0.250000000000001 |
| 0.199999999999999 | 0.199999999999999 |
| 0.166666666666667 | 0.166666666666667 |
| 0.142857142857143 | 0.142857142857143 |
| 0.125000000000000 | 0.125000000000000 |
| 0.111111111111111 | 0.111111111111111 |
| 0.100000000000000 | 0.100000000000000 |
| 2.928968253967685 | 0.000000000000000 |

The program produces printed output on file `nout`, if that parameter is positive. This is illustrated below, in which another least-squares problem (Ex. 21 in `minresqlpTestProgram`) is solved: $\min \|x\|$ such that $x \in \arg\min \| \operatorname{diag} \big[d, 0, 0\big] x - b\|$, where $d \equiv \big[\frac{1}{50}, \frac{2}{50}, \ldots, \frac{48}{50}\big]^T$ and $b \equiv \big[d, 1, 1\big]^T . * \big[50 : -1 : 3, 1, 1\big]^T$, where $.*$ indicates elementwise multiplication. No preconditioner is applied, and shift $\sigma = 0$.

Notice that the rightmost column of the 39th iteration is marked with "`P`", which indicates that the program switches from MINRES phase to MINRES-QLP phase since $\mathcal{A}_{39} \approx 1.81 \times 10^7 > \mathit{trancond} = 10^7$. Even though the last line in the output reports that MINRES-QLP has to stop at iteration 46 because $\|x_{47}\| > \mathit{maxxnorm}$, the algorithm appears to be successful because the relative error in $x_{46}$ is merely $2.8 \times 10^{-13}$.

```
Enter MINRES-QLP.      Solution of symmetric Ax = b
n = 50                 ||b|| = 6.78E+01          precon = F
itnlim = 200           rtol = 2.22E-16           shift = 0.00E+00
maxxnorm = 1.00E+07    Acondlim = 1.00E+15       trancond = 1.00E+07

iter            x(1)    xnorm    rnorm   Arnorm Compatible       LS  norm(A)  cond(A)
   0 0.0000000000E+00 0.00E+00 6.78E+01 3.69E+01   1.00E+00 1.00E+00 0.00E+00 1.00E+00
   1 1.7180943901E+00 1.16E+02 2.40E+01 1.09E+01   1.83E-01 6.94E-01 5.44E-01 1.00E+00
   2 3.8644538109E+00 1.53E+02 1.15E+01 4.58E+00   6.82E-02 6.06E-01 6.57E-01 1.70E+00
   3 6.3954779963E+00 1.72E+02 6.51E+00 2.30E+00   3.60E-02 5.37E-01 6.57E-01 2.27E+00
   4 9.2579303917E+00 1.83E+02 4.16E+00 1.29E+00   2.21E-02 4.74E-01 6.57E-01 2.94E+00
   5 1.2389816033E+01 1.90E+02 2.94E+00 7.91E-01   1.52E-02 4.10E-01 6.57E-01 3.74E+00
   6 1.5722893791E+01 1.95E+02 2.28E+00 5.14E-01   1.16E-02 3.43E-01 6.57E-01 4.78E+00
   7 1.9185796048E+01 2.00E+02 1.91E+00 3.50E-01   9.61E-03 2.79E-01 6.57E-01 6.20E+00
   8 2.2706980590E+01 2.03E+02 1.71E+00 2.48E-01   8.47E-03 2.21E-01 6.57E-01 8.20E+00
   9 2.6217158315E+01 2.07E+02 1.59E+00 1.81E-01   7.81E-03 1.73E-01 6.57E-01 1.10E+01

  10 2.9651001936E+01 2.10E+02 1.52E+00 1.36E-01   7.39E-03 1.36E-01 6.57E-01 1.50E+01
  20 4.9405101158E+01 2.71E+02 1.41E+00 1.08E-02   5.76E-03 1.16E-02 6.57E-01 1.92E+02
  30 4.9999971981E+01 3.22E+02 1.41E+00 6.37E-05   5.06E-03 6.86E-05 6.57E-01 1.18E+04
  39 5.0000000000E+01 3.53E+02 1.41E+00 2.09E-08   4.72E-03 2.25E-08 6.57E-01 1.81E+07 P

  40 5.0000000000E+01 3.56E+02 1.41E+00 6.60E-09   4.69E-03 7.10E-09 6.57E-01 5.29E+07
  45 5.0000000000E+01 3.76E+05 1.41E+00 5.53E-07   5.73E-06 5.95E-07 6.57E-01 2.01E+11
  46 5.0000000000E+01 2.07E+02 1.41E+00 5.86E-07   5.73E-06 6.30E-07 6.57E-01 2.01E+11

Exit MINRES-QLP.       istop = 12                itn = 46
Exit MINRES-QLP.       Anorm = 6.5701E-01        Acond = 2.0123E+11
Exit MINRES-QLP.       rnorm = 1.4142E+00        Arnorm = 1.2149E-05
Exit MINRES-QLP.       xnorm = 2.0717E+02
Exit MINRES-QLP.       xnorm has exceeded maxxnorm or will exceed it next iteration.
```

The items printed at the $k$th iteration are listed and explained in the source code. For simplicity we assumed no preconditioner below; when there is one, we simply replace $\bar{A}$ and $r_k$, respectively, with $\tilde{A}$ and $\bar{r}_k$ as defined in Section 3 and Algorithm 1.

Table V: Items printed at the $k$th iteration.

| **Label** | **Definition** |
|---|---|
| *iter* | The iteration number $k$. Results are always printed for the first 10 iterations and the last. Intermediate results are printed every 10th iteration. |
| $x(1)$ | The value of the first element of the approximate solution $x_k$. |
| *xnorm* | $\|x_k\|$. |
| *rnorm* | $\|r_k\|$. |
| *Arnorm* | $\|\bar{A}r_k\|$. |
| *Compatible* | A dimensionless quantity that should converge to zero if and only if $\bar{A}x = b$ is compatible. It is an estimate of $\|r_k\|/(\|\bar{A}\|\|x_k\| + \|b\|)$, which decreases monotonically. |
| *LS* | A dimensionless quantity that should converge to zero if and only if the optimum $r_k$ is nonzero. It is an estimate of $\|\bar{A}r_k\|/(\|\bar{A}\|\|r_k\|)$, which is usually not monotonic. |
| $norm(A)$ | A monotonically increasing underestimate of $\|\bar{A}\|$. |
| $cond(A)$ | A monotonically increasing underestimate of $\mathrm{cond}(\bar{A})$. |

The integer output `istop` takes an initial value of 0; when the program stops, it takes a positive integer value between 1 to 14 inclusive to signify one of the termination conditions in Table VI. We note that if `istop` $> 7$, the final $x$ may or may not be an acceptable solution. On the contrary, when `istop` $\leq 7$, we can be sure $x_k$ is a good or even an excellent approximate solution of a given problem.

Table VI: Termination conditions in MINRES-QLP.

| `istop` | **Termination Conditions** |
|---|---|
| 1 | $\beta_{k+1} < \varepsilon$. Iteration $k$ is the final Lanczos step. Rarely occurs. |
| 2 | $\beta_2 = 0$ in the Lanczos iteration; i.e. the second Lanczos vector is zero. This means the right-hand-side is very special. If there is no preconditioner, $b$ is an eigenvector of $A$. Otherwise (if *precon* is true), let $My = b$. If shift is zero, $y$ is a solution |

Table VI: Termination conditions in MINRES-QLP (continued).

| istop | **Termination Conditions** |
|---|---|
| | of the generalized eigenvalue problem $Ay = \lambda My$, with $\lambda = \alpha_1$ from the Lanczos vectors. In general, $(A - \sigma I)x = b$ has the solution $x = (1/\alpha_1)y$, where $My = b$. |
| 3 | $b = 0$, so the exact solution is $x = 0$. No iterations were performed. |
| 4 | $\lVert\bar{r}\rVert$ appears to be less than the value $rtol(\lVert\bar{A}\rVert\lVert\bar{x}\rVert + \lVert b\rVert)$; $x$ should be an acceptable solution of $\bar{A}x = b$. |
| 5 | $\lVert\bar{r}\rVert$ appears to be less than the value $\varepsilon(\lVert\bar{A}\rVert\lVert\bar{x}\rVert + \lVert b\rVert)$. This means that the solution is as accurate as seems reasonable on this machine. |
| 6 | $\lVert\bar{A}\bar{r}\rVert$ appears to be less than the value $rtol\lVert\bar{A}\rVert\lVert\bar{r}\rVert$; $x$ should be an acceptable least-squares solution. |
| 7 | $\lVert\bar{A}\bar{r}\rVert$ appears to be less than the value $\varepsilon\lVert\bar{A}\rVert\lVert\bar{r}\rVert$. This means that the least-squares solution is as accurate as seems reasonable on this machine. |
| 8 | The iteration limit was reached before convergence. |
| 9 | The matrix defined by `Aprod` does not appear to be symmetric. For certain vectors $y = Av$ and $r = Ay$, the products $y^Ty$ and $r^Tv$ differ significantly. |
| 10 | The matrix defined by `Msolve` does not appear to be symmetric. For vectors satisfying $My = v$ and $Mr = y$, the products $y^Ty$ and $r^Tv$ differ significantly. |
| 11 | An inner product of the form $x^TM^{-1}x$ was not positive, so the preconditioner $M$ does not appear to be positive definite. |
| 12 | $\lVert x\rVert$ has exceeded $maxxnorm$ or will exceed it next iteration. |
| 13 | $\mathrm{cond}(\bar{A})$ has exceeded $Acondlim$ or $0.1/\varepsilon$, so $\bar{A}$ must be very ill-conditioned. |
| 14 | $\lvert\gamma_k^{(4)}\rvert < \varepsilon$. This is probably a least-squares problem but residual norms have not satisfied NRBE conditions. |

## 6. AVAILABILITY

Implementations of MINRES-QLP are available in FORTRAN 90 and MATLAB 7.8 from the Systems Optimization Laboratory, Stanford University [SOL], or the first author's homepage `http://home.uchicago.edu/sctchoi/` under the terms of the OSI Common Public License (CPL) [OSI-CPL] or the BSD License [BSD].

ACKNOWLEDGMENTS

We thank Christopher Paige for his contribution to the theory of MINRES-QLP, including the simplified proof of Theorem 2.2. We also thank Tim Hopkins and David Saunders for testing and running our software on the NAG Fortran compiler and the Intel ifort compiler respectively, resulting in more robust code. We are grateful to Zhaojun Bai and two anonymous reviewers for their patience and constructive comments. The first author thanks Jed Brown, Ian Foster, Todd Munson, Gail Pieper, Stefan Wild, and Hong Zhang for their feedback, interest, and support for this work. Finally, we express our gratitude to the SIAM SIAG/LA 2012 Linear Algebra Prize Committee for their favorable consideration of MINRES-QLP.

REFERENCES

BLAS. Basic Linear Algebra Subprograms. http://www.netlib.org/blas/.

BSD. The BSD License. http://www.opensource.org/licenses/bsd-license.php.

Burkardt, J. Matrix Market Read and Write Utilities. http://people.sc.fsu.edu/~jburkardt/f_src/mm_io/mm_io.html.

Burkardt, J. 2012. fortran90 software. http://people.sc.fsu.edu/~jburkardt/.

Calvetti, D., Lewis, B., and Reichel, L. 2000. An L-curve for the MINRES method. In *Proceedings of SPIE*. Vol. 4116. 385–395.

Chivers, I. and Sleightholme, J. 2006. *Introduction to Programming with Fortran: With Coverage of Fortran 90, 95, 2003, and 77.* Springer-Verlag, London, UK.

Choi, S.-C. T. 2006. Iterative Methods for Singular Linear Equations and Least-Squares Problems. Ph.D. thesis, ICME, Stanford University, CA.

Choi, S.-C. T., Paige, C. C., and Saunders, M. A. 2011. MINRES-QLP: A Krylov subspace method for indefinite or singular symmetric systems. *SIAM J. Sci. Comput. 33,* 4, 1810–1836.

Davis, T. A. and Hu, Y. 2011. The University of Florida sparse matrix collection. *ACM Trans. Math. Software 38,* 1, 1:1–1:25.

Dongarra, J., Eijkhout, V., and Kalhan, A. 1995. Reverse communication interface for linear algebra templates for iterative methods. Report UT-CS-95-291, University of Tennessee, TN.

Fong, D. C.-L. 2011. Minimum-Residual Methods for Sparse Least-Squares Using Golub-Kahan Bidiagonalization. Ph.D. thesis, ICME, Stanford University, CA.

Fong, D. C.-L. and Saunders, M. A. 2011. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput. 33*, 2950–2971.

Fong, D. C.-L. and Saunders, M. A. 2012. CG versus MINRES: An empirical comparison. *SQU Journal for Science 17,* 1, 44–62.

Foster, L. 2009. San Jose State University Singular Matrix Database. http://www.math.sjsu.edu/singular/matrices/.

Freund, R. W. and Nachtigal, N. M. 1994. A new Krylov-subspace method for symmetric indefinite linear systems. In *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics*, W. F. Ames, Ed. IMACS, 1253–1256.

Golub, G. H. and Van Loan, C. F. 1996. *Matrix Computations*, 3rd ed. Johns Hopkins University Press, Baltimore, MD.

Hansen, P. C. 1998. *Rank-deficient and Discrete Ill-posed Problems.* SIAM Monographs on Mathematical Modeling and Computation. SIAM, Philadelphia, PA.

Hestenes, M. R. and Stiefel, E. 1952. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards 49*, 409–436.

Lanczos, C. 1950. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards 45*, 255–282.

Matrix Market. Text File Formats. http://math.nist.gov/MatrixMarket/formats.html.

O'Boyle, N. M., Tenderholt, A. L., and Langner, K. M. 2008. cclib: A library for package-independent computational chemistry algorithms. *J. Comput. Chem. 29,* 5, 839–845.

Oliveira, S. and Stewart, D. 2006. *Writing Scientific Software: A Guide to Good Style.* Cambridge University Press, Cambridge, UK.

OSI-CPL. Open Source Initiative (OSI) - Common Public License (CPL). http://www.opensource.org/licenses/cpl1.0.php.

Paige, C. C. 1976. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *J. Inst. Math. Appl. 18,* 3, 341–349.

Paige, C. C. and Saunders, M. A. 1975. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal. 12,* 4, 617–629.

Paige, C. C. and Saunders, M. A. 1982a. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software 8,* 1, 43–71.

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

Paige, C. C. and Strakoš, Z. 2002. Residual and backward error bounds in minimum residual Krylov subspace methods. *SIAM J. Sci. Comput. 23,* 6, 1899–1924.

Saad, Y. and Schultz, M. H. 1986. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput. 7,* 3, 856–869.

Simon, H. D. 1984. The Lanczos algorithm with partial reorthogonalization. *Math. Comp. 42,* 165, 115–142.

SOL. Systems Optimization Laboratory, Stanford University. www.stanford.edu/group/SOL.

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

Trefethen, L. N. and Bau, III, D. 1997. *Numerical Linear Algebra.* SIAM, Philadelphia, PA.

The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract DE-AC02-06CH11357 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

Argonne National Laboratory Preprint ANL/MCS-P3027-0812.