Complex LSQR and LSMR

Austin Benson and Victor Minden

{arbenson, vminden} @ stanford.edu

June 29, 2013

CME 338 / MS&E 318: Large Scale Numerical Optimization
Final Project
Instructor: Michael Saunders

1 Introduction

LSQR and LSMR are two popular methods for solving square and rectangular linear systems [1], [2]. Let A be an $m \times n$ matrix and b be an m-vector. Some of the problems that these methods can solve are

- $Ax = b$ when $m = n$
- $\min_x ||x||_2$ subject to $Ax = b$ when $m \leq n$
- $\min_x ||Ax - b||_2$ when $m \geq n$ (least squares)
- $\min_x ||Ax - b||_2^2 + ||\delta x||_2^2$

These methods are much more general than solvers such as CG, which operates on square, symmetric positive definite systems. Currently, the Systems Optimization Laboratory maintains code for LSQR and LSMR for A and b with *real* entries. However, it is desirable to solve problems when A and b have *complex* entries. For example, these problems arise in computational geophysics. LSQR and LSMR provide a sequence of iterates, x_k, that are approximate solutions to the problem. The principal property of x_k is that it lives in $\mathcal{K}_k(A^TA, A^Tb)$, the k-th Krylov subspace generated by A^TA and A^Tb.

For our project, we implemented complex LSQR and complex LSMR. Using properties of these algorithms, we can explain the differences in the code between the real and complex cases. We first give a brief overview of the Golub-Kahan bidiagonalization process, which is the first step of both LSQR and LSMR. We then describe the algorithms and the updates made to the software. This report includes many facts from the course notes [3].

Algorithm 1 Golub-Kahan bidiagonalization process

function GOLUB-KAHAN-BIDIAG(A, b)
  $\beta_1 = ||b||_2$
  $u_1 = b/\beta_1$
  $w = A^T u_1$
  $\alpha_1 = ||w||_2$
  $v_1 = w/\alpha_1$
  for $k = 1, 2, \ldots$ **do**
    $w = Av_k - \alpha_k u_k$
    $\beta_{k+1} = ||w||_2$
    $u_{k+1} = w/\beta_{k+1}$
    $w = A^T u_{k+1} - \beta_{k+1} v_k$
    $\alpha_{k+1} = ||w||_2$
    $v_{k+1} = w/\alpha_{k+1}$
  end for
end function

2 Golub-Kahan bidiagonalization process

Unlike Lanczos-based methods (CG, MINRES) and Arnoldi-based methods (GMRES), which are based on tridiagonal and upper Hessenberg systems, respectively, LSQR and LSMR generate iterates based on bidiagonal systems. The Golub-Kahan bidiagonlization algorithm is described in Algorithm 1. The algorithm produces a sequence of $(k+1) \times k$ bidiagonal matrices B_k:

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix}$$

We note that, even for complex A and b, B_k is a *real* matrix. This is precisely because all the entries are norms of vectors. This property will be important when discussing the implementation of complex LSQR and complex LSMR in the following sections. If A and b are complex, then the vectors u_k and v_k will also be complex. We note that $u_k \in \mathcal{K}_k(AA^T, b)$ and $v_k \in \mathcal{K}_k(A^T A, A^T b)$.

3 LSQR

From now on, we assume that A and b are complex. If the damping parameter δ is equal to 0, then the k-th iteration of LSQR solves the subproblem

$$\min_{y_k \in \mathbb{C}^k} ||B_k y_k - \beta_1 e_1||_2,$$

where e_1 is the first standard basis vector. If $\delta > 0$, then LSQR solves

$$\min_{y_k \in \mathbb{C}^k} \left|\left| \begin{pmatrix} B_k \\ \delta I \end{pmatrix} y_k - \begin{pmatrix} \beta_1 e_1 \\ 0 \end{pmatrix} \right|\right|_2$$

The k-th iterate of LSQR is $x_k = V_k y_k$, where the columns of V_k are $v_1, \ldots, v_k$ from the bidiagonalization of A. We note that LSQR is equivalent to CG on the normal equation

$$(A^T A + \delta^2 I)x = A^T b.$$

Since B_k and δ are real, y_k is also real. V_k is complex, so x_k is also complex. LSQR solves for y_k by performing a QR factorization of B_k or $\begin{pmatrix} B_k \\ \delta I \end{pmatrix}$. Since the linear system is real, we re-use code from the original LSQR software (for real A and b). However, the complex version of the LSQR code still required several updates:

1. The interface `zLSQR()` is provided for complex LSQR in the file `zlsqrModule.f90`.
2. The data types of the vectors u_k and v_k were changed to double complex vectors.
3. The black-box interfaces to the matrix-vector product functions for Av_k and $A^T u_k$ were changed so that a complex vector is returned from the function.
4. The complex BLAS routines `dznrm2` and `zscal` are included. These are used, for example, when updating u_k and v_k.
5. The fast norm estimates were updated were updated to account for the complex iterates x_k.

We updated the `Makefile` to make it easy to build the real and complex versions of LSQR. The commands are `make lsqr`, `make zlsqr`, and `make all` to build just the real version, just the complex version, and both real and complex versions, respectively. We also include tests for complex LSQR in the files `zlsqrTestProgram.f90` and `zlsqrTestModule.f90`. After running `make zlsqr`, the tests can be run with the command `./zTestProgram`.

4 LSMR

At the k-th iteration, LSMR solves the subproblem

$$\min_{y_k \in \mathbb{C}^k} \left\| \begin{pmatrix} R_k^T R_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{pmatrix} y_k - \alpha_1 \beta_1 e_1 \right\|,$$

where R_k is the upper triangular factor in the QR factorization of B_k or $\begin{pmatrix} B_k \\ \delta I \end{pmatrix}$. The iterate is then $x_k = V_k y_k$. Again, the system is real and is solved with a QR factorization. We note that LSMR is equivalent to MINRES on the normal equation

$$(A^T A + \delta^2 I)x = A^T b.$$

Since the system that is solved at each iteration is real, we re-use code from the original LSMR software. Again, several changes are still necessary for the complex version of LSMR, which we list here:

1. The interface `zLSMR()` is provided for complex LSMR in the file `zlsmrModule.f90`.

2. The data types of the vectors u_k and v_k were changed to double complex vectors.

3. The black-box interfaces to the matrix-vector product functions for Av_k and $A^T u_k$ were changed so that a complex vector is returned from the function.

4. The complex BLAS routines `dznrm2` and `zscal` are included.

We again updated the `Makefile` to make it easy to build the real and complex versions of LSMR. The commands are `make lsmr`, `make zlsmr`, and `make all`. We include tests in the files `zlsmrTestProgram.f90` and `zlsmrTestModule.f90`. After running `make zlsqr`, the tests can be run with the command `./zTestProgram`.

References

[1] 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

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

[3] Michael Saunders. Notes 3: Iterative Methods for Square and Rectangular Systems. *CME 338 / MS&E 318 course notes*. Spring 2013.
`http://www.stanford.edu/class/msande318/notes/notes03-unsymmetricCG.pdf`.