Math 310 Final Project
Recycled subspaces in the LSMR Krylov method

Victor Minden*

November 25, 2013

1 Introduction

In many applications, it is the case that we are interested not only in solving a single system $Ax = b$, but rather a sequence of systems with perhaps different but *related* system matrices or right-hand sides,

$$A^{(i)}x^{(i)} = b^{(i)}, \qquad i = 1, 2, ..., \tag{1}$$

where $A^{(i)} \in \mathbb{R}^{n\times n}$ and $b^{(i)}$ and $x^{(i)}$ are of appropriate size for all n. Such problems can be seen, for example, in the solution of PDEs with a time-stepping component, where the right-hand side depends on the solution to the previous system but the matrix perhaps remains fixed. More complicated examples where the system matrix varies as well can be imagined in cases such as solving smooth nonlinear systems of equations.

One class of methods for solving linear systems of equations are Krylov methods. Given a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $b \in \mathbb{R}^n$ (we do not consider the complex case here), the Krylov subspace of dimension k generated by A and b is given by

$$\mathcal{K}_k(A, b) \equiv \text{span}\{b, Ab, \ldots, A^{k-1}b\}, \tag{2}$$

where we note that $\mathcal{K}_1(A, b) \subset \mathcal{K}_2(A, b) \subset \cdots \subset \mathcal{K}_n(A, b)$. A Krylov method, then, is an iterative method that at each iteration bumps up the dimension of the Krylov subspace by one and looks for the optimal solution to the (perhaps ill-posed) problem $Ax = b$ within the new Krylov subspace, where optimal is an overloaded term that varies depending on the Krylov method employed.

One approach to solving $Ax = b$ is to solve the normal equations,

$$A^TAx = A^Tb. \tag{3}$$

Clearly this system is consistent and symmetric but is possibly singular. The LSMR method of Fong et al. [5] is a Krylov method that is equivalent to applying the well-known MINRES algorithm to (3), but in a way that is more numerically stable (as iterating on A^TA directly can lead to loss of accuracy due to the squaring of the condition number of A). One advantage of working with the normal equations is that the algorithm can solve not only $Ax = b$ for symmetric A but also least squares solutions for overdetermined systems.

A common criticism of iterative solvers such as Krylov methods when viewed in the context of solving (1) is that the method must begin anew with each $b^{(i)}$, so that each system is roughly as expensive as the first system to solve. In contrast, for sequences where the system matrix is fixed as $A^{(i)} = A$ for all i, direct methods generally pay an additional cost up front by factoring

*vminden@stanford.edu

the matrix, but then each new right-hand side is asymptotically less expensive to solve. As such, there is interest in determining a method of taking advantage of the relatedness of the successive systems when solving (1) with an iterative method, and in some way *recycling* information from one system to the next. This is the basic idea behind recycled Krylov methods, see, e.g., [13, 8]. In this document, we introduce a recycled version of the LSMR method, which we dub RLSMR.

2 Previous work

Parks et al. in [8] give a brief overview of two recycled Krylov subspace methods based on past ideas for improving the convergence of iterative methods using truncation or restarting. In GMRES, for example, the size of the space against which the new iterate must be orthogonalized grows with each iteration. Thus, in practice, it is common to restart GMRES after a certain number of iterations. To improve convergence, then, it is possible to deflate the GMRES iteration by using subspace information from previous restarts to remove components approximately along eigenvectors corresponding to small eigenvalues [7, 4]. The idea of recycling a Krylov subspace is similar and many of the same ideas can be employed, with the main difference being that now we are interested in solving multiple different systems as opposed to only reusing subspace information within a single linear solve.

In [13], Wang et al. consider the problem of subspace recycling in the vein of [8] in the case when the matrix is symmetric but not positive definite. For such systems, the Krylov method of choice is generally MINRES, which does not see the same problem of the length of the recursion growing with the number of iterations – as such, "restarting" MINRES doesn't hold any inherent appeal. However, when we are interested in solving many similar systems, then it could still be advantageous to use the recycling ideas from system to system in order to cut down on the required number of iterations on the whole sequence of systems, and it is this motivation that leads to the RMINRES algorithm.

We note that in [1], Baglama et al. explore the idea of an augmented LSQR method, which has at its core many ideas similar to that of recycling a Krylov method based on Golub-Kahan bidiagonalization. However, their motivation is different and they do not consider sequences of linear systems.

3 Background on LSMR

The LSMR iteration for (3) is built around the Golub-Kahan bidiagonalization procedure applied to the matrix A [11],

$$AV_k = U_{k+1}B_k, \tag{4}$$

$$A^T U_{k+1} = V_{k+1}L_{k+1}^T, \tag{5}$$

where L_k is lower bidiagonal and B_k is lower bidiagonal with one extra row, in fact containing L_k as a subblock:

$$B_k = \begin{pmatrix} L_k \\ \beta_{k+1}e_k^T \end{pmatrix}. \tag{6}$$

When initialized with $\beta_1 u_1 = b$ and $\alpha_1 v_1 = A^T u_1$, the columns of the matrix U_k span the Krylov subspace $\mathcal{K}_k(AA^T, b)$ and the columns of the matrix V_k span the Krylov subspace $\mathcal{K}_k(A^TA, A^Tb)$.

We note that, as derived in [5],

$$A^T A V_k = A^T U_{k+1} B_k \tag{7}$$
$$= V_{k+1} L_{k+1}^T B_k \tag{8}$$
$$= V_{k+1} \begin{pmatrix} B_k^T \\ \alpha_{k+1} e_{k+1}^T \end{pmatrix} B_k \tag{9}$$
$$= V_{k+1} \begin{pmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{pmatrix}, \tag{10}$$

where the β_ks and α_ks are the elements of B_k in accordance with the notation of Fong et al. Thus, the LSMR algorithm is theoretically equivalent to applying MINRES to (3). In the standard (non-recycled) case, this corresponds to searching at iteration k for the optimal least-squares solution $x_k \in \mathcal{K}_k(A^T A, A^T b)$, i.e.,

$$x_k = \underset{x \in \mathcal{K}_k}{\arg\min} \, ||A^T A x - A^T b||_2, \tag{11}$$

or equivalently,

$$x_k = V_k y_k, \tag{12}$$
$$y_k = \underset{y \in \mathbb{R}^k}{\arg\min} \, ||A^T A V_k y - A^T b||_2. \tag{13}$$

By using (7), we obtain

$$y_k = \underset{y \in \mathbb{R}^k}{\arg\min} \left|\left|A^T A V_k y - A^T b\right|\right|_2 \tag{14}$$
$$= \underset{y \in \mathbb{R}^k}{\arg\min} \left|\left| V_{k+1} \begin{pmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{pmatrix} y - A^T b \right|\right|_2 \tag{15}$$
$$= \underset{y \in \mathbb{R}^k}{\arg\min} \left|\left| V_{k+1} \begin{pmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{pmatrix} y - \alpha_1\beta_1 v_1 \right|\right|_2 \tag{16}$$
$$= \underset{y \in \mathbb{R}^k}{\arg\min} \left|\left| V_{k+1} \left[\begin{pmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{pmatrix} y - \alpha_1\beta_1 e_1 \right] \right|\right|_2 \tag{17}$$
$$= \underset{y \in \mathbb{R}^k}{\arg\min} \left|\left| \begin{pmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{pmatrix} y - \alpha_1\beta_1 e_1 \right|\right|_2, \tag{18}$$

where we can pull out the V_{k+1} factor since it has orthonormal columns [11].

4 RLSMR: A recycled LSMR variant

In accordance with past work, we define a *recycle space* for solving a linear system to simply be a subspace of the domain of A, i.e., the space in which x lives. Then, we use the space for recycling by simply removing it from the Krylov space generated by a Krylov method and solving separately for the piece of the solution in the recycle space.

Suppose that we are given a full rank matrix $Y \in \mathbb{R}^{n \times p}$ such that the range of Y, $\mathcal{R}(Y)$, is the desired recycle space. Then, we can use Y to find a matrix C such that the columns of C form an orthonormal basis for the *image of* Y *under* $A^T A$, as well as a matrix Z that is the preimage of C, i.e.,

$$C = A^T A Z, \tag{19}$$
$$C^T C = I. \tag{20}$$

A description of how to do this can be seen in Algorithm 1.

Result: C and Z such that $A^TAZ = C$ and $C^TC = I$, where C is a basis for the column space of A^TAY.
$[Q,R] \leftarrow$ reduced QR decomposition of A^TAY
$C \leftarrow Q$
$Z \leftarrow YR^{-1}$

Algorithm 1: Creating the recycle matrices [8]

For a recycled LSMR variant, we modify the Golub-Kahan process (4) such that the Krylov subspace for the solution is orthogonal to $\mathcal{R}(Z)$, i.e., the recycled space. One way of doing this is to consider bidiagonalizing $(b, A(I - CC^T))$ in the Golub-Kahan process

$$AV_k = U_{k+1}B_k, \tag{21}$$

$$(I - CC^T)A^TU_{k+1} = V_{k+1}L_{k+1}^T, \tag{22}$$

where we note that, at least theoretically, it is unnecessary to replace A with $A(I - CC^T)$ in the first line because V_k is orthogonal to $\mathcal{R}(C)$.

To then formulate LSMR with added recycling, we now have to look for the optimal least-squares solution restricted to a different space,

$$x_k \in \mathcal{K}_k\left((I - CC^T)A^TA(I - CC^T), (I - CC^T)A^Tb\right) \cup \mathcal{R}(Z), \tag{23}$$

or,

$$x_k = Zz_k + V_ky_k, \tag{24}$$

with y_k and z given such that x_k minimizes $||A^TAx - A^Tb||_2$. Using some algebra, we can write the residual norm in a form more amenable to computation, as derived below.

From $(I - CC^T)A^TU_{k+1} = V_{k+1}L_{k+1}^T$, we see that $A^TU_{k+1} = CC^TA^TU_{k+1} + V_{k+1}L_{k+1}^T$, and combining this with $AV_k = U_{k+1}B_k$ gives

$$A^TAV_k = A^TU_{k+1}B_k \tag{25}$$

$$= (CC^TA^TU_{k+1} + V_{k+1}L_{k+1}^T)B_k \tag{26}$$

$$= CC^TA^TU_{k+1}B_k + V_{k+1}\begin{pmatrix} B_k^TB_k \\ \alpha_{k+1}\beta_{k+1}e_k^T \end{pmatrix} \tag{27}$$

$$= CC^TA^TAV_k + V_{k+1}\begin{pmatrix} B_k^TB_k \\ \alpha_{k+1}\beta_{k+1}e_k^T \end{pmatrix}. \tag{28}$$

For notational convenience, define H_{k+1} as

$$H_{k+1} = \begin{pmatrix} B_k^TB_k \\ \alpha_{k+1}\beta_{k+1}e_k^T \end{pmatrix}. \tag{29}$$

Then we can write

$$A^TA\begin{pmatrix} Z & V_k \end{pmatrix} = \begin{pmatrix} C & V_{k+1} \end{pmatrix}\begin{pmatrix} I & C^TA^TAV_k \\ 0 & H_{k+1} \end{pmatrix}$$

We see that for $x = Zz + V_ky$,

$$||A^TAx - A^Tb||_2 = \left|\left|A^TA\begin{pmatrix} Z & V_k \end{pmatrix}\begin{pmatrix} z \\ y \end{pmatrix} - A^Tb\right|\right|_2 \tag{30}$$

$$= \left|\left|\begin{pmatrix} C & V_{k+1} \end{pmatrix}\begin{pmatrix} I & C^TA^TAV_k \\ 0 & H_{k+1} \end{pmatrix}\begin{pmatrix} z \\ y \end{pmatrix} - A^Tb\right|\right|_2 \tag{31}$$

$$= \left|\left|\begin{pmatrix} C & V_{k+1} \end{pmatrix}\left[\begin{pmatrix} I & C^TA^TAV_k \\ 0 & H_{k+1} \end{pmatrix}\begin{pmatrix} z \\ y \end{pmatrix} - \begin{pmatrix} C^TA^Tb \\ \alpha_1\beta_1e_1 \end{pmatrix}\right]\right|\right|_2, \tag{32}$$

and so we can pull out the matrices with orthogonal columns as before to see that the optimal z and y are given by

$$\begin{pmatrix} z_k \\ y_k \end{pmatrix} = \underset{(z,y)^T \in \mathbb{R}^{d+k}}{\arg\min} \left|\left| \begin{pmatrix} I & C^T A^T A V_k \\ 0 & H_{k+1} \end{pmatrix} \begin{pmatrix} z \\ y \end{pmatrix} - \begin{pmatrix} C^T A^T b \\ \alpha_1 \beta_1 e_1 \end{pmatrix} \right|\right|_2 \tag{33}$$

For each k, the above system must be solved, and thus we consider it the fundamental subproblem of the RLSMR method. We note that y_k can be found independently from z_k, since only the lower-right block is rectangular, and thus we can solve (33) as

$$y_k = \underset{y \in \mathbb{R}^k}{\arg\min} \left|\left| \begin{pmatrix} B_k^T B_k \\ \alpha_{k+1} \beta_{k+1} e_k^T \end{pmatrix} y - \alpha_1 \beta_1 e_1 \right|\right|_2, \tag{34}$$

$$z_k = C^T A^T b - C^T A^T A V_k y_k. \tag{35}$$

4.1 Solving the fundamental subproblem

We note that Fong et al. offer some efficient update algorithms for the LSMR solution vector such that the cost of each iteration does not change with the iteration count [5]. This holds in the case where reorthogonalization is not necessary, and is one of the features that makes LSMR attractive compared to GMRES. It is unclear that there exists an update method for the solution of the RLSMR subproblem due to the recycle-space contribution, and thus we consider only the case of one-sided reorthogonalization, where the cost of an iteration already increases with iteration index. It is also important to note that since we will be interested in the Ritz vectors of the resultant iteration later for the purposes of recycling, reorthogonalization is a natural step to take in order to retain accuracy.

In [5], the y_k equation of (34) is solved efficiently via a pair of QR decompositions, the details of which we don't go into in this writeup. We note that the QR decompositions can be updated in subsequent iterations in constant time, meaning the dominant cost of the algorithm for computing $V_k y_k$ is $\mathcal{O}(nk + k^2)$. Suffice to say, this is dominated by the cost of one-sided reorthogonalization, $\mathcal{O}(kn)$, since k is bounded by n.

Given y_k, we note that the Zz_k can be found from (34) in $\mathcal{O}(n(k+p))$ time. This is accomplished by storing $C^T A^T A$ before any iterations are performed and then at each iteration, once y_k is computed, performing the following sequence of operations (w here is a work vector) :

Result: Zz_k and $C^T A^T A v_k$, the solution component in the recycle space and the new stored intermediate vector
Compute $C^T A^T A v_k$
$w \leftarrow -C^T A^T A V_k y_k$
$w \leftarrow C^T A^T b - w$
$Zz_k \leftarrow Zw$

Algorithm 2: Computing the solution component in the recycle space

Note here that the first step costs $\mathcal{O}(pn)$ time, the second costs $\mathcal{O}(pk)$ time, the third costs $\mathcal{O}(k)$ time and the last is $\mathcal{O}(nk)$. Thus, computing this component of the solution costs $\mathcal{O}(n(k+p))$ time. Therefore, the *full* solution at each iteration can be computed in $\mathcal{O}(n(k+p))$ time, which is still on the same order as the cost of the one-sided reorthogonalization step in the Golub-Kahan process. We also note that unless we have a fast way to apply A and A^T (for example, A is sparse or can be applied as a Fourier operator or has some black-box representation), the dominant cost at each iteration overall is the application of A and A^T, which is in general $\mathcal{O}(n^2)$.

4.2 Computing the new recycle space

It remains to discuss how the recycle space is actually calculated during the course of a solve with RLSMR. We've outlined a strategy for how to perform iterations given a recycle space, but in solving

a sequence of linear systems we ideally would like to use the information gained during the Krylov iteration in order to generate a new recycle space for the next time around.

One way to do this, used by Wang et al. in [13], is to calculate the harmonic Ritz vectors corresponding to the p smallest Ritz values of the iteration matrix with respect to $\mathcal{R}(Z) \cup \mathcal{R}(V_s)$, where the columns of V_s are some subset of the vectors generated during the Golub-Kahan process. Wang et al. use a cycle framework similar to that used by Parks et al. in [8], wherein they let V_s be some number of the most recent vectors generated and then do multiple recomputations of the harmonic vectors per linear system solve. This is advantageous as it is a way of using all of the vectors v_i to contribute to the new recycle space, but does not involve solving a very large generalized eigenvalue problem.

We consider a different approach to selecting V_s. Rather than dealing with the full cycle framework used previously, we will run the core LSMR algorithm to convergence, say k iterations, since we are storing the V_k matrix anyway for re-orthogonalization. Then, we will choose the p vectors of V_k which correspond to the p largest magnitude entries of y_k. These vectors will be the columns of our matrix V_s. The rationale behind this choice is that we are looking to recycle the space that contains most of the solution, so we should look at both Z and the p vectors in the Krylov space which contain most of the solution restricted to that space.

So, we look at computing harmonic Ritz vectors of A^TA with respect to the space $\mathcal{R}(W)$ where $W = \begin{pmatrix} Z & V_s \end{pmatrix}$. This is equivalent [13] to solving the generalized eigenvalue problem

$$W^TA^TAA^TAWq = \theta W^TA^TAWq. \tag{36}$$

Let's consider the matrix A^TAW. Defining P_s to be the matrix which pulls out the columns of V_s from V_k, we know that

$$\begin{aligned} A^TAW &= A^TA\begin{pmatrix} Z & V_s \end{pmatrix} && (37)\\ &= \begin{pmatrix} C & A^TAV_s \end{pmatrix} && (38)\\ &= \begin{pmatrix} C & A^TAV_kP_s \end{pmatrix} && (39) \end{aligned}$$

We can then define the quantity $R = C^TA^TAV_kP_s$ which is something we accumulate during the LSMR iterations. With this, we see that $W^TA^TAA^TAW = (A^TAW)^T(A^TAW)$ is given in block form as

$$W^TA^TAA^TAW = \begin{pmatrix} I & R \\ R^T & R^TR \end{pmatrix}, \tag{40}$$

where the blocks cost $\mathcal{O}(p^2n)$ to compute. Additionally we need to look at the matrix W^TA^TAW which is given as

$$\begin{aligned} W^TA^TAW &= \begin{pmatrix} Z^TC & Z^TA^TAV_kP_s \\ P_s^TV_k^TC & P_s^TV_k^TA^TAV_kP_s \end{pmatrix} && (41)\\ &= \begin{pmatrix} Z^TC & (C^TV_k)P_s \\ P_s^T(V_k^TC) & P_s^TV_k^TA^TAV_kP_s \end{pmatrix} && (42)\\ &= \begin{pmatrix} Z^TC & 0 \\ 0 & P_s^TV_k^TA^TAV_kP_s \end{pmatrix}. && (43) \end{aligned}$$

These blocks can be computed in $\mathcal{O}(p^2n)$ time.

After computing these matrices, the results can be sent to an eigensolver, which will yield the Ritz values and vectors in $\mathcal{O}(p^3)$ time. The approximate eigenvectors can then be recovered via left multiplication of the Ritz vectors by W [13].

5 Numerical experiments

6 Test descriptions

6.0.1 Random vector

For our first test for RLSMR, we generate a random matrix with Gaussian i.i.d. entries, G. Then, we slightly shift its spectrum to improve conditioning for our numerical tests, forming the matrix $A = G + sI$, where $s \geq 0$ is a parameter. To obtain a sequence of related systems, we consider forming b, a random vector with Gaussian i.i.d. entries, and then solving the sequence of problems

$$Ax^{(i)} = b + w_i, \tag{44}$$

where w_i is a Gaussian i.i.d. random vector with variance $\sigma_i^2 = 2^{-2i}$. The intuition here is that we have a true right-hand side, b, and then we allow the right-hand side to converge to it in successive iterations, much like we might see in an optimization routine (though we admit an optimization routine probably wouldn't face this idealized of a situation).

6.0.2 Random matrix

For a second test, we consider the same situation as above, but A is now allowed to vary and b is a fixed right hand side. We generate a random matrix with Gaussian i.i.d. entries, G, and perturb as before to form A. Then we form b, a random vector with Gaussian i.i.d. entries. Finally, we consider solving the sequence of problems

$$A^{(i)}x^{(i)} = b, \tag{45}$$

where $A^{(i)} = A + B_i$, where B_i is an i.i.d. Gaussian random matrix with variance $\sigma_i^2 = 2^{-2i}$.

6.0.3 Heat equation

As a final test for the RLSMR algorithm, we consider applying it to a simple one-dimensional heat equation with Neumann boundary conditions. Consider solving

$$u_t = u_{xx} \qquad x \in \Omega, \tag{46}$$

$$u_x = 0 \qquad x \in \partial\Omega, \tag{47}$$

$$u(x, 0) = u_0(x) \tag{48}$$

where $\Omega = [0, 1]$. We will use a simple finite difference scheme to discretize the PDE. Suppose we use an N-point discretization of the unit interval. It is simple to verify that a Crank-Nicholson discretization in time and a standard central difference in space gives the following discretized form of the PDE:

$$(I - \frac{\Delta t}{2\Delta x^2}L)U^{t+1} = (I + \frac{\Delta t}{2\Delta x^2}L)U^t, \tag{49}$$

where $L \in \mathbb{R}^{N\times N}$ is the standard tridiagonal finite-difference Laplacian operator except for the $(1, 2)$ and $(N, N - 1)$ entries, which are both equal to 2.

Because the heat equation is a simple evolution equation of an initial distribution u_0, we might expect that the solution in successive time steps would be similar enough that it might be possible to gleam information from the subspaces acquired during previous iterations. Thus, we consider solving a number of timesteps of this system, reusing subspaces as we go.

We note that for this application, the matrix happens to already be symmetric and positive definite, so neither LSMR nor RLSMR is really the method of choice, but this should provide a fair metric for the two.

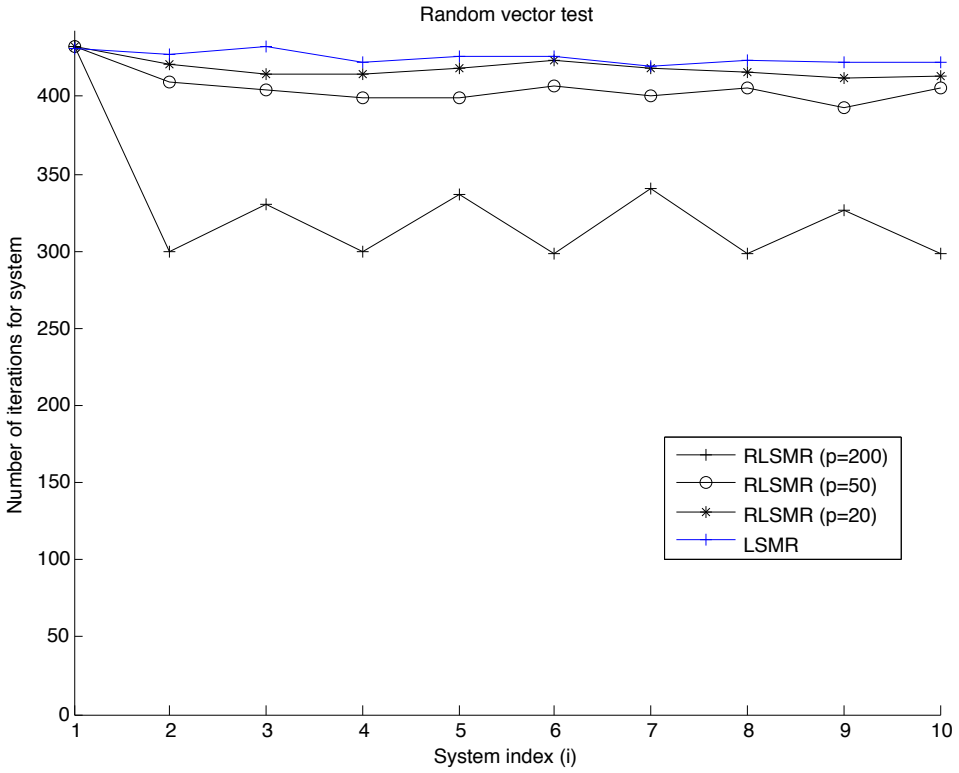

Figure 1: Iterations versus index for random vector test.

6.1 Results

In all numerical tests, we choose the stopping rule suggested by Fong et al. for consistent systems [5],

$$||Ax - b||_2 \leq \texttt{atol}||A||_2 \cdot ||x||_2 + \texttt{btol}||b||_2. \tag{50}$$

Since x is not known, we use the current estimate of x. We choose $\texttt{atol} = \texttt{btol} = \texttt{tol} = 1e-12$. All tests are on square systems of size $n = 1000$. For the two random matrix cases, we chose a shift parameter of $s = 10$, leading to matrices with condition numbers on the order of 1000.

6.1.1 Random vector

In Figure 1, we see how the number of iterations required to solve the system settles down as we change the right-hand side. The curves correspond to recycle spaces of dimension $p = 200$, $p = 50$, and $p = 20$, as well as the classical LSMR algorithm with no recycling. Full one-sided reorthogonalization was used on the matrix V_k.

6.1.2 Random matrix

Figure 2 is analogous to the previous figure, but corresponds to changing the system matrix itself instead of the right hand side. The curves correspond to recycle spaces of dimension $p = 200$, $p = 50$, and $p = 20$, as well as the classical LSMR algorithm with no recycling. Once again, full one-sided reorthogonalization was used on the matrix V_k. It is worth noting that the condition number of

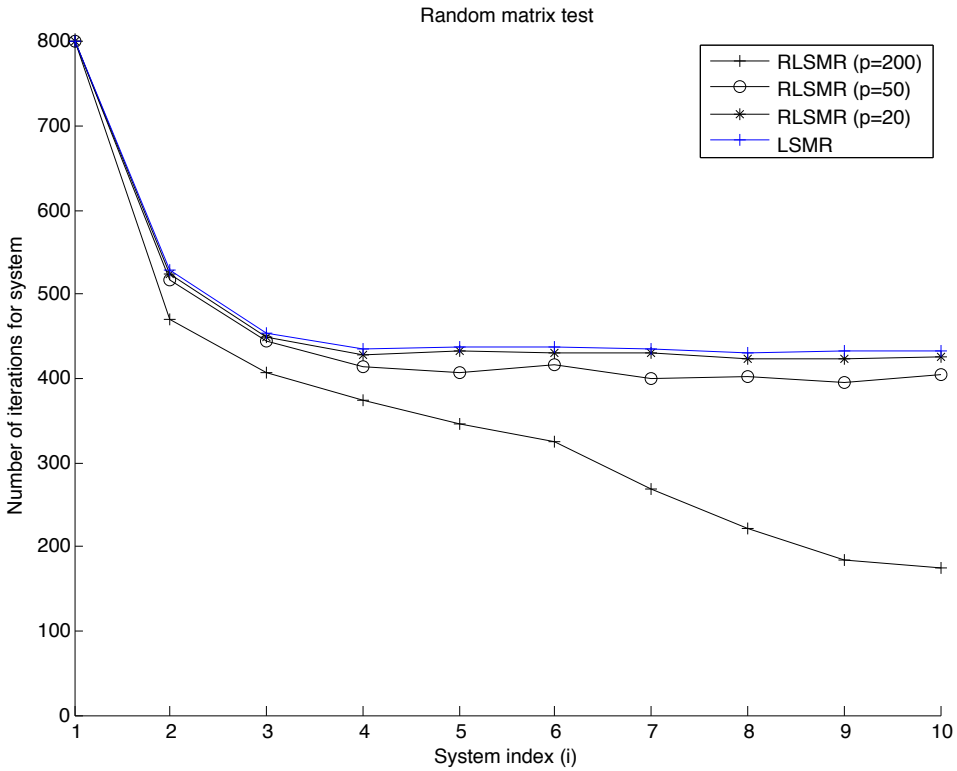

Figure 2: Iterations versus index for random matrix test. Note that the expected condition number of the system is monotonically decreasing with the system index, accounting for the shape of the LSMR curve.

the system changes in this example, with the expected condition number decreasing monotonically, which is why the number of iterations for LSMR decreases with system index.

6.1.3 Heat equation

Finally, Figure 3 shows the results of 10 time steps of the heat equation, once again with recycle spaces of dimension $p = 200$, $p = 50$, and $p = 20$. The time step size chosen for this example was $\frac{\Delta t}{2\Delta x^2} = 5$, though we note that the Crank-Nicholson scheme used is unconditionally stable.

7 Discussion

We see in Figure 1 that LSMR consistently takes around 430 iterations to converge to the desired tolerance of $1e-12$. Using a small number of recycle vectors lowers the iteration count a little bit, but it's only really when we reach $p = 200$ or so that the number of required iterations is significantly lower. This seems to indicate that there is merit in the idea of recycling subspaces, though the oscillatory nature of the curve for $p = 200$ is worrisome. One possible explanation is a feedback loop: when a large number of iterations is used, the harmonic Ritz vectors are more accurate, so the new recycle space is better, so fewer iterations are used, so the Ritz vectors are less accurate, so the new recycle space is worse, so more iterations are used.

In Figure 2, the situation is more favorable. Again, we don't really see a marked benefit until

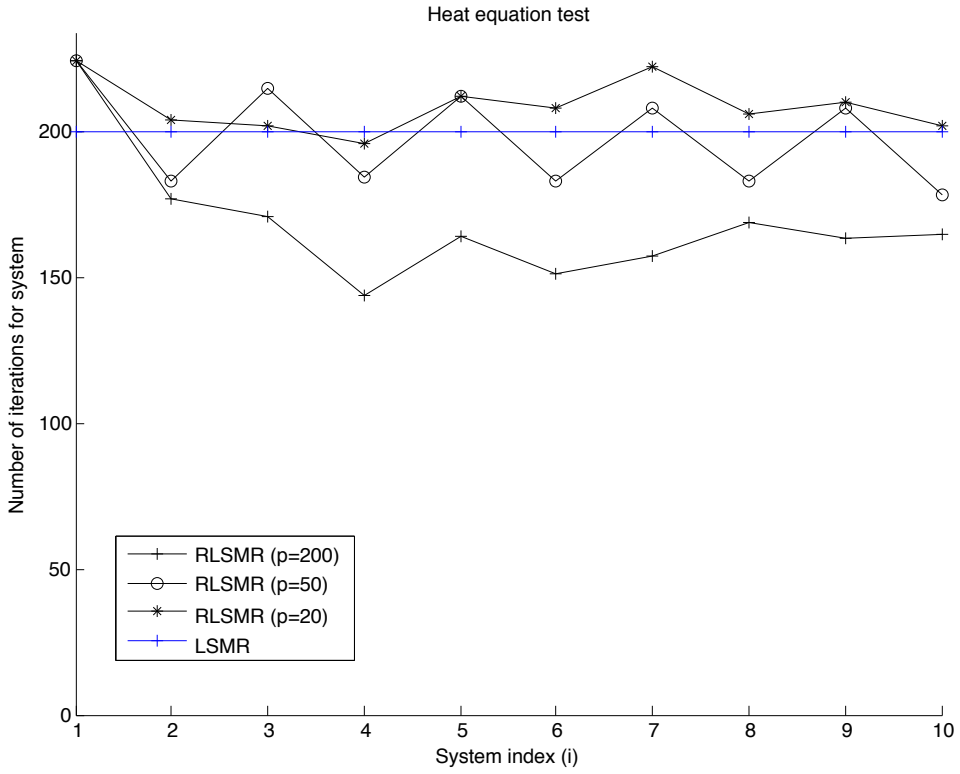

Figure 3: Iterations versus index for heat equation test.

the number of recycled vectors is relatively large at $p = 200$, but fewer than half the iterations are required there when compared to LSMR for later problems, which is a substantial benefit.

Finally, in Figure 3 we see once again oscillatory behavior in the iteration count for all choices of p. Unfortunately, in a number of cases the RLSMR versions of the algorithm perform *worse* than the LSMR version without recycling. This was unexpected but is quite possible – RLSMR and LSMR generate different Krylov spaces.

One thing that the different tests showed is that it is hard to intuit what changes to the system will lead to Krylov spaces that are spectrally similar. It was expected that time-stepping the heat-equation with a small time step would see the most direct gain from recycling, since the system is not changing and the right-hand side is only varying slowly with successive iterations. However, we saw that this was not the case.

There are an number of disadvantages to the RLSMR approach. The first, as mentioned previously, is that we do not get anything for free. By using a recycle space, the fast update formulas for the solution vector derived by Fong et al. for LSMR [5] can no longer be used, which means that we must now actually solve a (structured) system at each iteration to find the new solution vector. One could argue that the fast update algorithms are really the heart of LSMR (or even most Krylov methods, bar GMRES) and thus this is quite a high price to pay. However, in practice, if we are already performing full one-sided reorthogonalization, nothing we do here is asymptotically more expensive.

Another disadvantage of RLSMR is that it is unclear that the harmonic Ritz vectors lead to good eigenvector estimates in many cases. For example, if the system converges in a small number of iterations, then we are limited to computing Ritz vectors with respect to a low-dimensional subspace, which probably ruins our recycle space for future approximations. As such, some method

is needed to assess the utility or validity of a recycle space.

It is clear from the numerical experiments that *in some cases*, RLSMR can converge in many fewer iterations than LSMR. This could be an attractive quality in large-scale applications especially in the case of parallelization, where fewer iterations means fewer global communications in the case of Krylov methods. However, for this benefit, it would be necessary to look at cases where reorthogonalization is not necessary, in which case it is probable that the speed of standard LSMR iterations will outweigh the benefit.

In examples where the right-hand side is changing but the system matrix is not, it seems likely that direct methods are still preferable to recycled Krylov methods such as LSMR, as exploratory analysis in MATLAB showed that the backslash command beats both LSMR and RLSMR by orders of magnitude as far as timing goes. However, in applications where the system matrix itself is changing, Figure 2 seems to imply there's at least a chance that recycled methods could be of utility.

8 Conclusions

We introduced the recycled Krylov subspace framework to the LSMR method of Fong et al. [5], providing a new recycled subspace algorithm based on the Golub-Kahan bidiagonalization scheme, RLSMR. To our knowledge, this has not previously been done. We constructed a MATLAB implementation of the new algorithm and tested the implementation on a few toy examples and analyzed its strengths and weaknesses as compared to LSMR without recycling.

We saw that the performance of RLSMR is somewhat inconsistent but that there is definite merit to the idea of subspace recycling. Wang et al. [13] in their RMINRES algorithm do a thorough job of preconditioning and otherwise making the system as amenable to recycling as possible, with the result being that they seem to see a greater decrease in iterations from using recycled subspace information. In the future, it would perhaps be beneficial to investigate these techniques in greater detail on a specific problem in order to truly gauge the performance of RLSMR. Nonetheless, we believe the RLSMR algorithm is at least an interesting theoretical contribution to recycled subspace theory.

References

[1] J. Baglama, L. Reichel, and D. Richmond, *An augmented lsqr method*, Numerical Algorithms, 64 (2013), pp. 263–293.

[2] P. Benner and L. Feng, *Recycling Krylov subspaces for solving linear systems with successively changing right-hand sides arising in model reduction*, in Model Reduction for Circuit Simulation, P. Benner, M. Hinze, and E. J. W. ter Maten, eds., vol. 74 of Lecture Notes in Electrical Engineering, Springer Netherlands, 2011, pp. 125–140.

[3] B. A. Cipra, *The Best of the 20th Century: Editors Name Top 10 Algorithms*, SIAM News, 33 (2000).

[4] E. de Sturler, *Truncation strategies for optimal Krylov subspace methods*, SIAM J. Numer. Anal, 36 (1999), pp. 864–889.

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

[6] M. R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[7] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.

[8] M. L. Parks, E. D. Sturler, G. Mackey, D. D. Johnson, and S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, tech. rep., SIAM J. Sci. Comput, 2004.

[9] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd ed., 2003.

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

[11] M. Saunders, *CME 338 notes: Iterative methods for square and rectangular systems.* `http://www.stanford.edu/class/msande318/notes/notes03-unsymmetricCG.pdf`, 2013.

[12] ——, *CME 338 notes: Iterative methods for symmetric Ax=b.* `http://www.stanford.edu/class/msande318/notes/notes02-symmetricCG.pdf`, 2013.

[13] S. Wang, E. d. Sturler, and G. H. Paulino, *Large-scale topology optimization using preconditioned Krylov subspace methods with recycling*, International Journal for Numerical Methods in Engineering, 69 (2007), pp. 2441–2468.