Enhanced Preconditioners for Large Sparse Least Squares Problems *

Yousef Saad † Maria Sosonkina ‡

January 18, 2001

Abstract

Solving the normal equation systems arising from least-squares problems can be challenging because these systems are often very ill-conditioned. Though the use of iterative methods is appealing for very large problems of this type, preconditioning remains a stumbling block. In this paper we address this issue and describe a number of preconditioning strategies. In particular, we discuss a technique of finding a diagonal shift for the normal equations in order to achieve a good convergence rate. Numerical experiments are described to test the proposed strategies.

1 Introduction

Large sparse least-squares problems are gaining importance across many engineering and scientific disciplines. One often cited example is that of interior point methods in linear programming which require solving a succession of large sparse least-squares problems. It is commonplace to solve these least-squares systems with direct techniques based on QR factorizations [3]. However, relatively little attention has been devoted to their solution by iterative methods, in spite of the appeal of such methods for very large problems.

Part of the difficulty in using iterative methods is the dearth of effective preconditioning strategies. Since the normal equations tend to be ill-conditioned, preconditioning is perhaps even more important than for standard linear systems. Yet, it is also because these problems are ill-conditioned that it is hard to precondition them. In this paper we address this issue.

We consider techniques for solving least-squares problems of the form:

$$\min_x \|b - Ax\|_2, \tag{1}$$

*This work was supported in part by a grant from the Army Research Office under grant DAAD19-00-1-0485 and in part by the Minnesota Supercomputer Institute

†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455, saad@cs.umn.edu.

‡Department of Computer Science, University of Minnesota - Duluth, 320 Heller Hall, 10 University Drive, Duluth, MN 55812-2496. masha@d.umn.edu

where A is a sparse $n \times m$ matrix, with $n > m$. The solution of the above system can be obtained by solving the normal equations

$$A^T Ax = A^T b, \tag{2}$$

The above system is referred to as the system of normal equations of the Normal Residual (NR) type. Iterative methods can be used for solving the above system but they are known to experience difficulties since the condition number of the coefficient matrix $B = A^T A$, is often very large since it is the square of that of A (see, e.g., [3, 17]).

Another class of normal equations is embodied by the system

$$AA^T y = b, \mathrm{x} = A^T y \tag{3}$$

which is targeted at overdetermined systems, i.e., to the case $n < m$. The system (3) is often dubbed a 'Normal Error' (NE) type system and the methods developed for the NR systems can easily adapted to solving NE systems.

Preconditioning methods are essential if one is to solve the linear system (2) successfully by an iterative method. In contrast with the problem of solving standard linear systems, relatively few methods have been developed in this context. Such methods include the class of row (or column) projection techniques, which are equivalent to relaxation procedures (Gauss-Seidel, SSOR) applied to the system of normal equations (2) or (3). The Incomplete Cholesky (IC) factorization technique applied to the matrix $B = A^T A$ has also been advocated. However, this approach faces two distinct difficulties which have hampered its use.

First, the IC preconditioner, like all ILU preconditioners, is known to exist only for M-matrices. The positive definiteness of the matrix $B = A^T A$ is not sufficient to guarantee the existence of a factorization – or when a factorization exists, that it is of a good quality. When applied directly to (2), the IC factorization algorithm may fail by producing negative or zero values for diagonal entries of the factor L. As early as in 1978, Kershaw suggested this approach and replaced occurences of negative L_{ii} by an arbitrary positive element [11]. Kershaw stated that this simple remedy was effective for the class of problems he was considering and that it was in fact seldom required to resort to the remedy.

The second difficulty is that the matrix B can be very ill conditioned and this could yield a preconditioner that is 'unstable', meaning that the norm of L^{-1} can be huge, leading to an ineffective preconditioner. A possible remedy to both problems, is to shift the matrix B by a scalar α prior to computing its IC factorization. Thus, a preconditioner for $B' = B + \alpha I$ is constructed and applied to the original (nonshifted) matrix B. This approach was suggested by several authors, see, e.g., [13]. For large α, the preconditioner is obviously inaccurate. For smaller values of α on the other hand, the preconditioner, when it exists, does represent B more accurately but the norm of its inverse may become large, or very large. A compromise is therefore needed to obtain a shift α such that the corresponding preconditioner is stable and, at the same time, accurately represents the matrix.

One of the goals of this paper is to present strategies for selecting good shift values α and to study the dependency of the quality of preconditioning and the convergence rate on α. A possible option that is considered, is to start with a 'standard' choice of α and attempt to 'correct' the preconditioner. The correction procedure takes place in the preconditioner

application phase and is performed by means of rational approximation as described in [8]. Specifically, a rational approximation of B^{-1} is constructed by exploiting an expansion in terms of $(B + \alpha I)^{-i}$.

The paper is organized as follows. Section 2 gives an overview of existing preconditioning techniques for normal equations. The preprocessing steps such as scaling and reordering are explained in Section 3 followed by Section 4 dealing with matrix shifting strategies. Section 4 contains detailed descriptions of shifted incomplete LQ and Cholesky factorizations followed by a few heuristics for selecting an appropriate shift value. In section 5, we note on our usage of the rational approximation in preconditioner application. Numerical experiments are presented in Section 6 followed by the summary of the work.

2 Preconditioners based on A^TA or AA^T

An obvious technique for solving (2) is to form the matrix $B = A^TA$ and then obtain some incomplete factorization of B which is then used in conjunction with the accelerator CGNR [17]. We discuss only the NR option here, i.e., normal equations of the form (2). Preconditioners for the NE option (3) can be developed similarly.

Once the matrix B is available, it can be approximately factored by some incomplete factorization procedure. The simplest of these is the incomplete Cholesky without fill-in, or IC(0). As was indicated above IC(0) does not necessarily exist for positive definite matrices and therefore one can expect it to fail when computing that of B. In addition, even when such a factorization exists, its quality may be very poor.

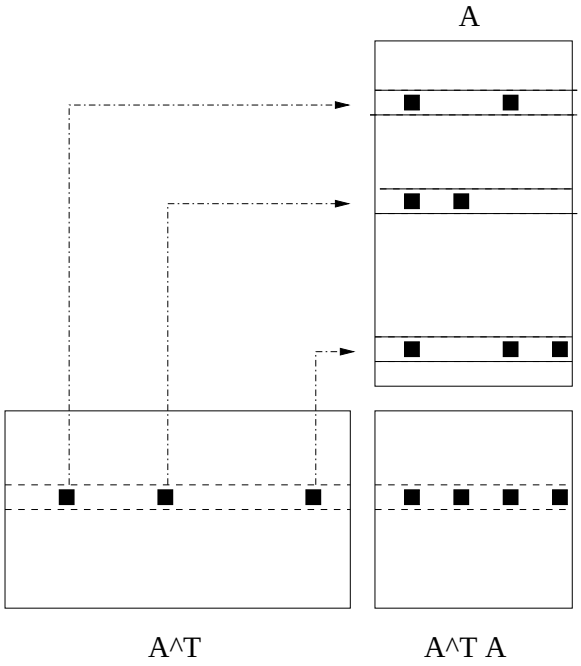
An accurate IC factorization similar to the threshold-based ILUT [16], can also be performed. In particular, one such factorization developed by Chow [4] uses Gaussian elimination with threshold and exploits an implementation proposed first by Jones and Plassman [10]. This technique was used successfully for solving saddle point problems in [12] and [14].

A drawback in using a technique based on an incomplete factorization of $B = A^TA$ is that this matrix must be formed. While this may not be a problem in many cases, there are situations in which B is far from being sparse. In fact B will generally be much less sparse than A itself. For example, a single dense row in A will yield a completely dense matrix B.

Fortunately, the algorithms ILUT and ICT do not require the whole matrix to be available at the same time. In fact the IKJ implementations of these algorithms compute one row of L (and the corresponding row of U in the case of ILUT) at the main step i same time. These variants require only one row of the matrix B at each of the elimination steps. If the structures of A and A^T are available at the same time, then an arbitrary row of B can easily be computed, then used to obtain the corresponding row of the factor L and then discarded. The construction of an arbitrary row of B is illustrated in Figure 1. This requires the row structure of both A and A^T. In certain implementations, a "companion linked list data structure" is employed to avoid having to construct and store the transpose [15]. This entails storing pointers to the entries of the transpose without restoring the values and saves some storage, though there are also advantages in storing the tranpsose in case it is to be used many times.

A number of other methods have been developed which can also avoid explicitly keeping

Figure 1: Construction of a row of B from the row structures of A and A^T



$B = A^T A$ in a similar way. One of these techniques is the Incomplete LQ (ILQ) factorization described in [15], which will discussed in Section 4.1.

3 Scaling and reordering strategies

It is sometimes helpful to preprocess the least-squares system before attempting to solve it by an iterative procedure. Preprocessing consists of a combination of techniques whose goal is to make the system better conditioned or to reduce fill-in in the incomplete factorization. We first discuss scaling.

Scaling of rows or columns of linear systems often helps improve its condition number. For solving least-squares systems, only the columns can be scaled without changing the solution. Instead of (1), we can solve

$$\min_y \|b - ADy\|_2, \qquad x = Dy \tag{4}$$

where D is a diagonal matrix of dimension m. The j-th entry of the diaginal D scales the j-th column of A. The corresponding system of normal equations becomes,

$$DA^T ADy = DA^T b, \tag{5}$$

If D is chosen to scale the columns of A to have 2-norm unity, then the coefficient matrix of the scaled normal equations (5) has diagonal entries equal to unity. This can be viewed as a diagonal preconditioning on the normal equations.

On the other hand, scaling of the rows of the least-squares system changes the solution. Indeed, the following system

$$\min_y \|D(b - Ax)\|_2, \tag{6}$$

clearly yields a least-squares solution using a different norm than the usual 2-norm.

4 Shifting Strategies

When preconditionning the matrix $B = A^T A$ by an incomplete Cholesky (IC) factorization, two distinct difficulties may be encountered. The first is that the IC factorization may not be defined – because the algorithm for constructing the factor L encounters a nonpositive diagonal element. This was noted by Kershaw [11]. The second difficulty is that, assuming the factorization itself does not cause a problem, the factor L obtained may be extremely ill-conditioned. Indeed, L may be an 'unstable' factor, which means that L^{-1} can be very large. Note that this can happen even when the residual matrix,

$$R = B - LL^T \tag{7}$$

is reasonably small. When preconditioning the original system with this matrix, the residual matrix R is magnified causing the CG accelerator to fail. Column pivoting may be used to help but this remedy distroys the matrix stucture, and experience with ILUT [17] in the nonsymmetric case suggests that it does not solve all problems.

The simplest way to 'stabilize' the IC factors is to perform the factorization on a perturbed B matrix. In its simplest form, the perturbation consists of using a constant shift which may be expressed as $B + \alpha I$. A preconditioner for

$$B' = B + \alpha I \tag{8}$$

is then constructed and applied to the nonshifted matrix B. The only difference with the standard IC algorithm for A^T is that a scalar is added to the diagonal entry when this is computed.

4.1 Shifted ILQ

As is well-known, incomplete LQ factorizations via Gram-Schmidt are mathematically equivalent to certain forms of Incomplete Cholesky factorizations for the matrix $B = A^T A$. Therefore one should expect that a simple strategy exists for incorporating shifts into ILQ factorizations as well.

To take into account the shift value in an incomplete LQ factorization, the following equality is used:

$$B + \alpha I = [A, \sqrt{\alpha} I]^T [A, \sqrt{\alpha} I], \tag{9}$$

where $A' = [A, \sqrt{\alpha} I]$ is the matrix A to which the $m \times m$ matrix αI is appended. Thus A' has dimensions $(n+m) \times m$. An incomplete LQ preconditioner is then applied to the matrix A'. For the sake of completeness, we recall here the Incomplete Modified Gram-Schmidt algorithm, see [15, 17] for details.

In the exact case, the problem is to compute a lower triangular matrix L and an orthogonal matrix Q such that $A = LQ$. This can be achieved in a number of different ways. We can, for example, exploit the fact that the Cholesky factor of the matrix $B = AA^T$ is identical with the matrix L of the decomposition $A = LQ$. This requires forming the data structure of the matrix AA^T, which may be much denser than A, but reordering techniques can be used to reduce the amount of work required to compute L. We will refer to this as *symmetric squaring.*

Another way of proceeding is to use a Gram-Schmidt process. This approach which may seem undesirable at first was successfully used in the past. Since the rows remain very sparse in the process, a given row of A will be typically orthogonal to most of the previous rows of Q, making the Gram-Schmidt process much less prone to numerical difficulties.

The method is based on using the classical (or modified) Gram-Schmidt process and incorporating a *dropping* strategy, to drop small elements according to a certain rule, in order to avoid excessive fill-in. In the incomplete LQ factorization described here, dropping is based on the magnitude of the elements generated. Similarly to the ILU factorization with dropping, the simplest idea is to keep the p_L largest elements in a row of L and the p_Q largest elements in a row of Q, where p_L and p_Q are two chosen parameters. The corresponding general procedure is as follows:

ALGORITHM **4.1 Incomplete LQ factorization**

1. *For* $i = 1, 2, ..., N$ *Do:*
2. *If* $i = 1$ *define* $\hat{q}_1 = a_1$ *and goto 5.*
3. *Compute all nonzero inner products* $l_{ij} = q_j a_i^T, j = 1, 2, ..., i-1$.
4. *Set* $l_{ij} = 0$ *if* l_{ij} *is not among the largest largest* l_{ij}*'s* $j = 1, 2, ..., i-1$
5. *Compute* $\hat{q}_i = a_i - \sum_{j=1, l_{ij} \neq 0}^{j=i-1} l_{ij} q_j$
6. *Keep only the* p_Q *largest elements of* $\hat{q}_i$ *(i.e., assign a zero value to all other elements)*
7. *Compute* $l_{ii} = \|\hat{q}_i\|_2$ *and* $q_i = \hat{q}_i / l_{ii}$
8. *EndDo*

The computation of all inner products $q_1 a_i^T, q_2 a_i^T, ..., q_{i-1} a_i^T$, must be done carefully in order to avoid an $O(n^2)$ cost. Since most of these inner products are equal to zero by sparsity, it is possible to compute them inexpensively. The observation, made in [15], is that if we call l the column of the $i-1$ inner products l_{ij} then l is the product of the matrix Q_{i-1} whose rows are $q_1, .., q_{i-1}$ by the vector a_i^T, i.e.,

$$l = Q_{i-1} a_i^T \tag{10}$$

This is the product of a sparse matrix by a sparse vector, which is best computed in 'sparse-sparse' mode, if the column data structure of Q_{i-1} is available. The method proposed in [15] uses this strategy. A linked list companion data structure for the columns of Q_i is computed and updated as each new row of Q becomes available.

4.2 Shifted IC(0)

We now go back to the use of shifts for IC(0). Clearly, the larger the shift the more stable the resulting preconditioner. However, the preconditioner error R increases. Figure 2 depicts

the change in the stability of IC(0) relative to its accuracy as α changes. The figure suggests that some optimal value of α must exist. The stability of the preconditioner is related to the norm of $\|L^{-1}\|_1$ whereas the accuracy is measured by the $\|R\|_1$. We estimate these norms as $\|L^{-1}e\|_\infty$ and $\|Re\|_\infty$, respectively. The factorization IC(0) is rather inexpensive to compute so we may compute a few factorizations IC(0) and use this information to determine an optimal value of α.

An appealing strategy is to use only the curve of $\|L_\alpha^{-1}e\|$ to determine a good value for α. For example, let

$$f(\alpha) = \|L_\alpha^{-1}e\|_\infty \tag{11}$$

At infinity, $f(\alpha) = 0$, and the function is likely to be fairly well represented by a rational approximation of the form

$$f(\alpha) \approx \frac{1}{\beta + \delta\alpha} \tag{12}$$

It is possible to fit the data with a few points. In the simplest case we could use only two values α_1, α_2 from which we could then trivially extract δ and β. Once β and δ are known, the question remains as to which α to select. Using the curve of Figure 2 for illustration, one possible strategy is to take the leftmost α before $f(\alpha)$ starts moving too fast as α varies. In practice this means that the critical point α_c is such that the derivative of f at this point equals a preset negative value $-s$:

$$-\frac{\delta}{(\beta + \delta\alpha_c)^2} = -s,$$

which leads to

$$\alpha_c = \frac{1}{\delta}\left[\sqrt{\frac{\delta}{s}} - \beta\right] \tag{13}$$

Note that the rightmost pole of the rational function $f(\alpha)$ may be located at the left or the right of the origin. The value of β will then be positive or negative respectively. On the other hand, δ should always be positive, reflecting a decreasing function for large enough α. Thus, formula (13) does not exclude a value of α_c that is negative, which actually takes place when $\beta > \sqrt{\delta/s}$. The two plots on Figure 2 illustrate the two situations: it is likely that the pole is negative for the plot on the left side, whereas it seems to be positive for the right figure.

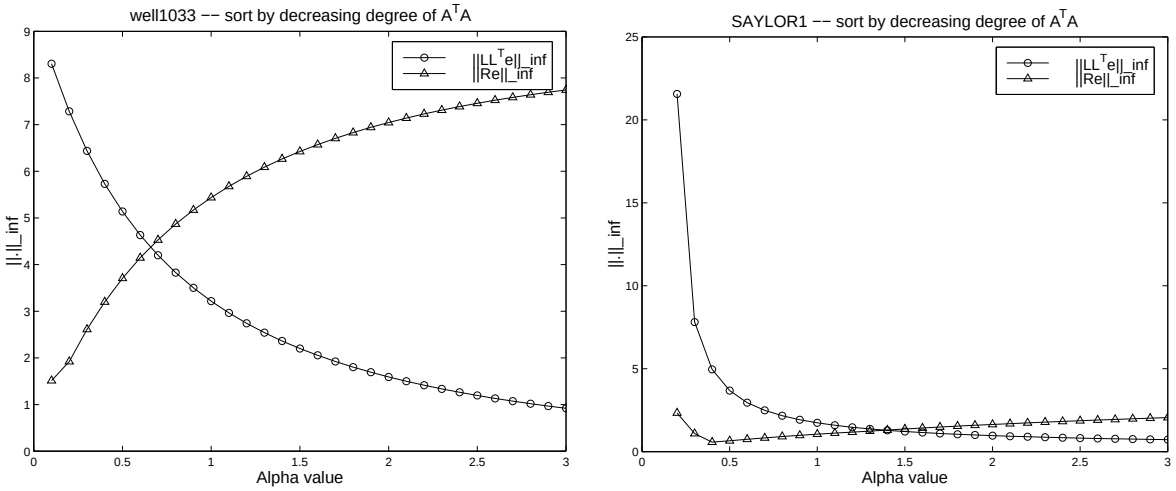
A related strategy is based on yet another least-squares data fitting method with a rational function. This strategy goes at the heart of the problem and attempts to approximate the function

$$h(\alpha) = \|I - L_\alpha^{-1}BL_\alpha^{-T}\|_2 \tag{14}$$

for a certain matrix norm. If L_α is the exact Cholesky factor of B, the above function should be zero for $\alpha = 0$. Since the factorization is costly to compute for any given α, we proceed similarly to the previous technique by replacing it with the function:

$$h_v(\alpha) = \|(I - L_\alpha^{-1}BL_\alpha^{-T})v\|_2 \tag{15}$$

Figure 2: Stability and accuracy of IC(0) factorization versus the shift value



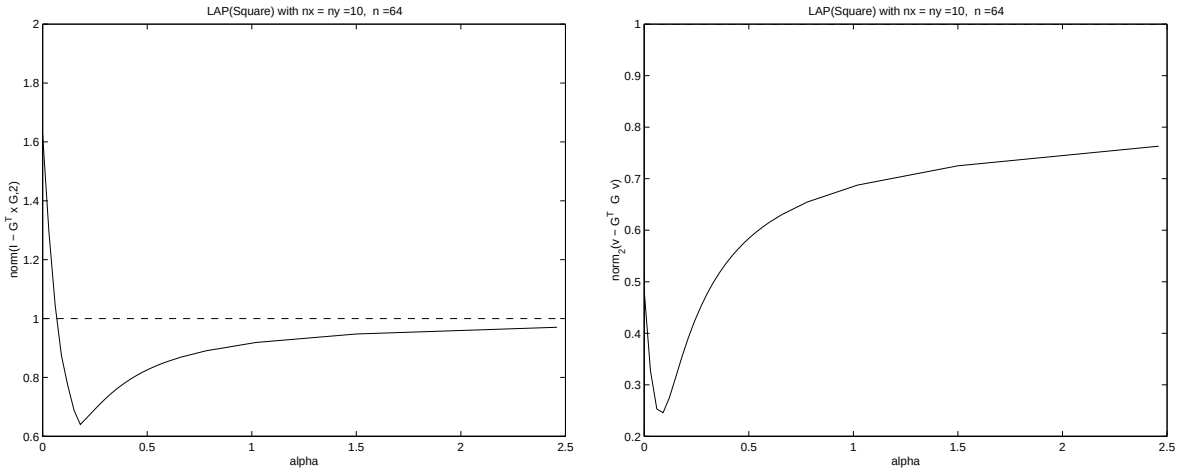
where v is a normalized trial vector. The vector v can be taken as a random vector or the vector of all ones, or a vector selected to attempt to yield a large norm for $L_\alpha^{-T}v$ as is done in the techniques used for estimating condition numbers (see, e.g., [7]).

An illustration of the behavior of the function $h(\alpha)$ is shown in Figure 3. The curve was generated from the following MatLab code:

```
gr = numgrid('S',10);      %%  generates a 10 x 10 cent. diff. grid
A = delsq(gr);             %%  generates the Laplacean matrix for this grid
B = A'*A;
n = size(B,1)
alph = 0.0 ;
dalph = 0.03;
i = 0;
while (alph < 5.0)
    i = i+1;
    B1 = B + alph * speye(n,n);
    R1 = cholinc(B1,'0') ;
    G = A*inv(R1);
    x(i) = alph;
    ff(i) = norm(full(speye(n) - G'*G),2) ;
    alph = alph + dalph;
    if (alph > 0.6)
        dalph = dalph*2;
    end
end
```

The asymptotic behavior of the above function is clear. Indeed, for large α the incomplete Cholesky factorization of $B + \alpha I$ will become very close to its exact Cholesky factorization.

Figure 3: The error measure $h(\alpha)$ and its estimate for a small Laplacean problem



As a result, $G_\alpha^T G_\alpha$ will approach $B + \alpha I$ and the eigenvalues of $I - L_\alpha^{-1} B L_\alpha^{-T}$ will become close to $\alpha/(\alpha + \lambda_i)$ whose maximum (with respect to i) approaches 1, as α increases. The general shape of the curves shown in 3 for large values of α is easily understood from the above argument. Near the origin both curves see the measure of the preconditioning error increase, though the estimate using a trial vector does not increase as much. What is common in both cases is a steep dip of the measure before it reaches a minimum and starts increasing again toward the asymptotic value.

Because of the asymptotic behavior of $h(\alpha)$, a steep dip followed by a move up to converge to a constant, it may be rather difficult to capture the behavior of the function $h(\alpha)$ as a function of α for difficult problems. For example, the brief dip may easily missed or poorly represented by the estimate. As a result, we have opted to 'scale' the function h, and its estimate h_v by the norm of L_α:

$$g(\alpha) = \|I - L_\alpha^{-1} B L_\alpha^{-T}\| \; \|L_\alpha L_\alpha^T\| \tag{16}$$

The justification for this is as follows. Note that L_α is the exact Cholesky factor for the matrix $B_\alpha + \alpha I - R_\alpha$, where R_α is the matrix of dropped entries during the incomplete factorization. Therefore,

$$\begin{aligned} I - L_\alpha^{-1} B L_\alpha^{-T} &= I - L_\alpha^{-1}(B + \alpha I - R_\alpha) L_\alpha^{-T} + L_\alpha^{-1}(\alpha I - R_\alpha) L_\alpha^{-T} \\ &= L_\alpha^{-1} \left(\alpha I - R_\alpha\right) L_\alpha^{-T} \end{aligned}$$

As was noted above, for large values of α, the residual matrix R_α is nearly equal to zero, in which case the function above behaves like $\alpha \|L_\alpha^{-1} L_\alpha^{-T}\|$, which should converge to a constant. After multiplying by the norm of $L_\alpha L_\alpha^T$ the resulting function behaves like $\alpha\kappa(L_\alpha L_\alpha^T)$ for large values of α. Without this scaling, it may be the case that the minimum value found from the approximation to h_v would be reached for very large values of α. For very small values, the influence of this second factor should be moderate. Thus, for large values of α the function $h_v(\alpha)$ should increase almost linearly with α. The rest of the behavior around zero (which

is the important one) is not too different from that of h_v. In reality, the function used for interpolating $g_{1,v}$ replaces the norm of $L_\alpha L_\alpha^T$ by its maximum diagonal entry.

For the reasons indicated above the function h_v is approximated by a rational function of the form:

$$\tilde{g}(t) = \frac{\delta_1 + \delta_2 * t + \delta_3 * t^2}{\delta_4 + t} \tag{17}$$

The curve is approximated in the least-squares sense as follows. If t_i, g_i are the measured values for the points, then we would solve in the least squares sense the following equations:

$$\delta_1 + \delta_2 * t_i + \delta_3 * t_i^2 = g_i(\delta_4 + t_i)$$

or

$$\delta_1 + \delta_2 * t_i + \delta_3 * t_i^2 - g_i * \delta_4 = g_i t_i$$

The minimum of the approximation $\tilde{g}$ is then computed by setting its derivative to zero. The details are omitted. This minimal value is used as the best alpha.

The selection of the nodes for data fitting is important. We need a first value α_0 that is far enough to be on the asymptotic regime of f, yet not too far in order not to loose accuracy on the representation of f. The next point α_1 is selected to be $\alpha_0/2$. Additional points can be added by successive halving. If the procedure breaks down too soon, leading to a non-positive diagonal entry in L before enough data points are available to yield an approximation of the rational function, the α's are started again to the right of α_0, this time by doubling the shift after each loop.

4.3 Use of a condition number estimator

A constant value of α may be chosen by examining the matrix B, one row at a time. If column scaling is assumed, then we can select for example,

$$\alpha = \max_i \|b_i'\|_2, \tag{18}$$

where i $(i = 1, \ldots, m)$ is a row number and b_i' is the part of the ith row of matrix B with the column indices less than i. The resulting value of α may be large so the preconditioner stability is ensured. However, the shift estimate may also be too conservative since it is applied to the *entire* diagonal even in the case when the matrix has only few nondiagonally dominant rows.

Since an incomplete factorization is constructed one row at a time, we can design a procedure of estimating α separately for each row. Thus, instead of equation 8, we now have

$$B = A + D, \tag{19}$$

where D is some diagonal matrix. For this purpose, we consider an incremental condition number estimator. In particular, an estimator suggested in [1] suits well our goals of changing α as factorization proceeds. Implemented in [5], the incremental condition number estimation (`laic1`) is a $\mathcal{O}(n)$ procedure for estimating the smallest or the largest singular value of a lower triangular matrix L as it being constructed starting with row 1. Assume that we want to

find an estimate for the smallest singular value $\hat{\sigma}$. Given the current estimates σ and the corresponding singular vector w as well as the last row of L and the last diagonal entry γ in L, the algorithm finds $\hat{w} = (sw, c)^T$ such that $\hat{w}$ is an appropriate singular vector of L such that $\|L\hat{w}\|_2 = \hat{\sigma}$ and the pair $(s, c)^T$ and $\hat{\sigma}$ is an eigenpair of the system

$$\begin{pmatrix} \sigma^2 & \\ & 0 \end{pmatrix} + \begin{pmatrix} \theta \\ \gamma \end{pmatrix} (\theta, \gamma)^T, \quad \theta = w^T x.$$

This procedure may be easily adapted for the sparse matrix computations such that it uses the number of row nonzeros $\mathcal{O}(nnz_i)$ operations. One such a modification has been described in [2]. A sparse version of condition estimation can be used after each row of the factorization is computed (see Algorithm 4.2).

ALGORITHM **4.2** *Shifting Strategy with condition number estimation*

1. *Start with* $\alpha_0 \neq 0$
2. *In the end of factorization for row* i *(*i*=2,...,*m*),*
3. *Compare condition number* κ_i *with a pre-set value* t*:*
3. *If less than* t *then*
4. *Reduce* α
5. *else Increase* α*,*
6. *Augment current (*i*th) diagonal element*
7. *Recompute* κ_i
8. *endif*
9. *Continue factorization.*

Once a tool for assessing the stability of an incomplete factorization "on the fly" is provided, we can devise and compare several strategies for choosing shift value for *each* row of the original matrix. In particular, we have tested α increase (line 5) with the maximum norm over rows of L computed so far (i.e., for rows 1 to i). Such a flexibility is affordable since a condition number estimate may be updated in $\mathcal{O}(1)$ operations (line 7) when the ith (the last computed) row of the factorization is modified. Specifically, if the current diagonal entry of L is changed, we call the `laic1` algorithm again but with modified γ and 'old' θ, i.e., θ is already known from the previous call to the estimator. In line 4, the reduction of α may be also carried out in many ways. For example, the closer the end of the factorization, the more aggressive is the decrease in α. Note that the reduction in α affects the construction of the *next* row of the triangular factor.

5 Improving accuracy via rational approximation

A technique presented in [8] deals with the problem of improving the preconditioning of a shifted matrix, by means of an extrapolation process based on rational approximation. The framework of the method is that of an arbitrary (square) matrix B which is shifted into $C = B + \alpha I$ and then approximately factored. It was argued in [8] that this factorization is in general inadequate and that an extrapolation step can help improve convergence dramatically.

Table 1: Test problem descriptions

Name	n	m	$n_z(A)$	$n_z(A^TA)$	Matrix source
well1033	1,033	320	4,732	3,974	Geodesy
illc1033	1,033	320	4,732	3,974	Geodesy
lanimalS	28,254	17,263	128,763	75,002	Animal breeding
lanimalB	56,508	34,526	523,912	225,006	Animal breeding

Specifically, let us assume that the Cholesky factor L_α of $B + \alpha I$ is computed. Then, instead of using $M_\alpha = L_\alpha L_\alpha^T$ as a preconditioner to B, the method would use the rational expansion:

$$M_{\alpha,d} = \sum_{i=1}^{d} \alpha^{i-1} M_\alpha^i \tag{20}$$

where d is a small integer (typically < 10) representing the degree of the rational expansion.

This is based on the rational expansion of $1/\lambda$ around α:

$$\frac{1}{\lambda} = \sum_{i=1}^{d} \frac{\alpha^{i-1}}{(\lambda+\alpha)^i}$$

If the error in an incomplete Cholesky factorization is large, then more accuracy can be achieved by the expansion

$$M_{\alpha,d} = \sum_{i=0}^{d-1} (I - AM_\alpha^{-1})^i, \tag{21}$$

which is equivalent to equation (20) when M_α is the exact factorization of A. See [8] for a detailed comparison of equations (20) and (21).

6 Numerical Experiments

The goal of the numerical experiments presented here is to compare the convergence rates and times for different strategies for selecting the shift α. The least-squares problems for the experiments have been taken from Harwell-Boeing collection [6]. The animal breeding problems are from the CERFACS collection [9]. The problem characteristics are provided in Table 1. The columns labeled n, m, $n_z(A)$, and $n_z(A^TA)$ state the row dimensions, column dimensions, number of nonzeros in A, and number of nonzeros in A^TA, respectively. Problems `illc1033` and `lanimalB` are more difficult to solve than `well1033` and `lanimalS`, respectively, due to ill-conditioning.

The experiments were performed on a PC with a Pentium II processor and 128 MB of RAM. We have used the conjugate gradient algorithm for normal equations to solve the test problems. The rational approximation was implemented as in equation (21) since it usually gives a more accurate approximation in the case of large factorization error. In Table 2, the columns have the following meanings:

Table 2: Comparison of various strategies for selecting α

		α_{strD}	α_{strS}	α_f	α_g	$\alpha = 0$
well1033-d4	Iter.	51	47	43	50	38
	Prec.	.03	.02	.11	.19	.02
	Sol.	.27	.28	.26	.29	.23
illc1033-d4	Iter.	971	831	1000	576	†
	Sol.	5.10	5.09	6.09	3.53	†
lanimalS-d2	Iter.	133	68	570	85	68
	Prec.	9.55	.82	3.65	5.47	.62
	Sol.	28.37	14.76	122.87	18.65	14.76
lanimalB-d2	Iter.	124	562	174	228	†
	Prec.	47.31	6.53	32.20	42.79	†
	Sol.	83.25	401.30	124.78	163.63	†

Column name	Strategy for choosing α
α_{strD}	as in Algorithm 4.2
α_{strS}	as in equation (18)
α_f	as in approximation for the function $f(\alpha)$
α_g	as in approximation for the function $g(\alpha)$

The block-row names state the problem name suffixed with the degree of the rational approximation used to solve this problem. For example, `well1033-d4` denotes the problem `well1033` and degree of approximation 4. The iteration numbers, preconditioner construction time, and the solution time are shown in rows `Iter.`, `Prec.`, and `Sol.`, respectively. Note that the preconditioner construction includes the α selection time.

In strategies α_{strD}, α_f, and α_g, the starting choice of α was 0.005 for all the tests except for the `ill1033` and `lanimalB` problems, for which strategies α_f and α_g started with α selected by α_{strS}.

Table 2 shows that for the large problems requiring careful selection of α (cf. results for `lanimalB-d2`) the strategies which attempt to arrive at some 'optimal' α are superior to the conservative α selection or the case without shifting. On the other hand for easy test problems, rows `lanimalS-d2` and `well1033-d4`, the best α seems to be zero, so the strategy without shift performs the best.

In the ILQ preconditioner enhanced with a shifting strategy (say α_{strS}), we vary the fill-in in both L and Q along with the degree of rational approximation (Column `(Deg., Fill)`) of Table 3). It appears that a high degree speeds up convergence and enables memory savings. Thus the solution time (Column `Sol.`) is lower than when no rational approximation is used despite the increased expense of the preconditioning operation.

For the matrix `lanimalB`, we have plotted the measured values of $g(\alpha)$ (equation (16)) against the shift values (Figure 4) and have fitted the function $\tilde{g}$ (dashed-dotted line) into the data. One may observe that $g(\alpha)$ is well approximated by the fitting function.

Table 3: Dependence of fill-in on the degree of approximation for the ILQ preconditioning

Name	(Deg., Fill)	n_z	Prec.	Sol.	Iter.
well1033	(0, 30)	5,120	.05	.17	99
	(2, 20)	2,740	.03	.26	76
	(4, 10)	1,236	.01	.38	84
illc1033	(0, 30)	5,146	.05	1.68	1,000
	(2, 20)	3,304	.03	3.66	1,000
	(4, 10)	1,344	.02	1.07	230
lanimalS	(0, 30)	457,231	4.26	29.41	201
	(2, 20)	293,401	2.11	32.80	106
	(4, 10)	145,757	.80	28.97	78

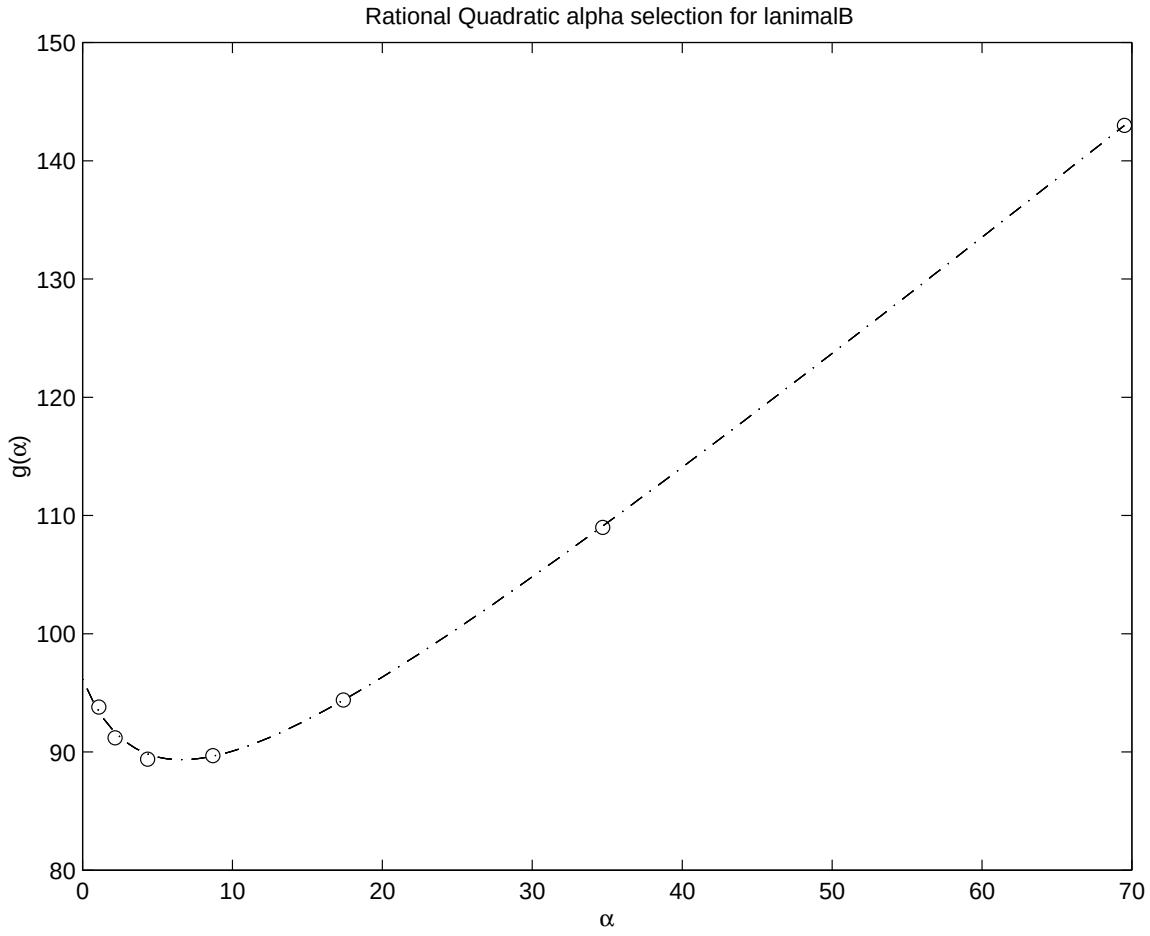


Figure 4: Approximation of the $g(\alpha)$ with a rational function

References

[1] C. H. Bischof. Incremental condition estimation. *SIAM J. Matrix Anal. Appl*, 11:312–322, 1990.

[2] C. H. Bischof, J. G. Lewis, and D. J. Pierce. Incremental condition estimation for sparse matrices. *SIAM J. Matrix Anal. Appl*, 11:644–659, 1990.

[3] A. Bjork. *Numerical Methods for Least-Squares Problems*. SIAM publications, Philadelphia, PA, 1996.

[4] E. Chow. Personal Communication, 1999.

[5] J. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, and D. Sorensen. Prospectus for the development of a linear algebra library for high-performance computers. Technical Report Tech. Mem. No. 97, Argonne National Lab, 1987.

[6] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15:1–14, 1989.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, Second edition, 1989.

[8] P. Guillaume, Y. Saad, and M. Sosonkina. Rational approximation preconditioners for general sparse linear systems. Technical Report umsi-99-209, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1999.

[9] M. Hegland. Description and use of animal breeding data for least squares problems. Technical Report TR/PA/93/50, ETH-Zurich, 1993.

[10] M. Jones and P. Plassman. An improved incomplete choleski factorization. *ACM Transactions on Mathematical Software*, 21:5–17, 1995.

[11] D. Kershaw. Solution of single tridiagonal systems and vectorization of the ICCG algorithm on the CRAY-1. In Garry Rodrigue, editor, *Parallel Computations*, pages 85–89. Academic Press, 1982.

[12] L. Little and Y. Saad. Block LU preconditioners for symmetric and nonsymmetric saddle point problems. Technical Report umsi-99-104, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1999.

[13] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computations*, 34:473–497, 1980.

[14] I. Perugia, V. Simoncini, and M. Arioli. Linear algebra methods in a mixed approximation of magnetostatic problems. Technical Report 1060, IAN-CNR, 1997.

[15] Y. Saad. Preconditioning techniques for indefinite and nonsymmetric linear systems. *Journal of Computational and Applied Mathematics*, 24:89–105, 1988.

[16] Y. Saad. ILUT: a dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994.

[17] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS publishing, New York, 1996.