Rational Approximation Preconditioners for General Sparse Linear Systems

Philippe Guillaume * Yousef Saad † Maria Sosonkina ‡

November 10, 1999

Abstract

This paper presents a class of preconditioning techniques which exploit rational function approximations to the original matrix. The matrix is first shifted and then an incomplete LU factorization of the resulting matrix is computed. The resulting factors are then used to compute a better preconditioner to the original matrix. Since the incomplete factorization is made on a shifted matrix, a good LU factorization is obtained without allowing much fill-in. The result needs to be extrapolated to the non-shifted matrix. Thus, the main motivation for this process is to save memory. The method is useful for matrices whose incomplete LU factorizations are poor, e.g., unstable. An error analysis for the conjugate gradient algorithm gives some guidance for choosing the shift of the matrix, in the special case where the shifted system is solved exactly.

1 Introduction

Rational approximation preconditioners are targeted at extremely ill-conditioned linear systems. Examples of such systems are those that arise in the modeling of thin Shells. These systems tend to be very difficult to solve by iterative methods despite the fact that they are symmetric positive definite. The problem is that the quality of incomplete LU (in this case Cholesky) factorizations for such matrices can be so poor that they become ineffective. It is tempting to simply shift the matrix A by a scalar α and extract the preconditioning for $A+\alpha I$ which is then used for preconditioning the original matrix, see, e.g., [7]. This by itself does not work well enough in general.

A modification of this idea which involves more work, will lead to a more effective technique. This modification consists of exploiting a rational approximation to A^{-1} based on an expansion in terms of the form $(A+\alpha I)^{-i}$. Because the matrix is shifted, its LU factorization might be a fairly accurate factorization of $A+\alpha I$. More importantly, it is more

*UMR MIP 5640, Département de Mathématiques, INSA, Complexe Scientifique de Rangueil, 31077 Toulouse Cedex, France, guillaum@gmm.insa-tlse.fr

†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455, saad@cs.umn.edu. This work was supported in part by NSF under grant CCR-9618827, and in part by the Minnesota Supercomputer Institute

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

likely to be stable. In fact, instability is the main problem with incomplete LU factorization preconditioners of very ill-conditioned matrices. Thus, the idea is to still manage to solve such problems by using a fairly inaccurate factorization (less fill-in) obtained from $A + \alpha I$.

In the special case where the shifted system is solved exactly, some error bounds are obtained for the conjugate gradient algorithm. A first error bound describes the behavior of the approximate solution at the beginning of the iteration process, and explains the fast decay of the error observed at the first steps in numerical experiments. A second error bound shows that the rational transformation strongly improves the rate of convergence at the asymptotic regime, where, for example, a large accuracy is required.

The next section introduces the rational preconditioner based on the (incomplete) LU factorization of $A+\alpha I$, and presents some natural extensions to other types of perturbations of the original matrix. Section 3 analyses the error in the case of an exact LU factorization of $A + \alpha I$, and discusses how to choose the shift α of the matrix. Finally, some numerical experiments with solving difficult real-world problems in structural mechanics are reported in Section 4.

2 Rational Approximation Preconditioners

Consider the linear system

$$Ax = b,$$

where A is a non singular square matrix of dimension n. A number of iterative methods approximate the solution $x = A^{-1}b$ to the above system, by a vector of the form

$$\tilde{x} = p(A)b.$$

where p is a certain polynomial. The approximation theory underlying these methods is to approximate the rational function $s(\lambda) = 1/\lambda$ by a polynomial $p(\lambda)$ of degree d. The approximation is to be accurate on the (discrete) set of eigenvalues of A. However, polynomial approximation preconditioners have their limitations, and as a result they are currently seldom used.

Rational approximations can be considered as an alternative. The first reaction to this possible approach is that the problem may not be well defined, since the best approximation to $1/\lambda$ by rational functions is $1/\lambda$ itself. A hint at a possible approach is to consider a similar situation that is naturally encountered when solving large eigenvalue problems. In shift-and-invert strategies, [9], it is common to compute an eigenvalue λ_i by using a Krylov-subspace type method on the matrix $(A - \sigma I)^{-1}$, where σ is chosen to be close to the desired eigenvalue λ_i. More general rational Krylov subspaces have also been used in [10] for eigenvalue computation. The goal is similar here since the inverse function is to be approximated by a rational function with a pole close to the origin.

2.1 Approximations to $1/\lambda$

We wish to obtain the best possible approximation to the function $s(\lambda) \equiv 1/\lambda$ from an expansion of the form

$$\frac{1}{\lambda} \simeq \eta_0 + \frac{\eta_1}{\lambda+\alpha} + \frac{\eta_2}{(\lambda+\alpha)^2} + \ldots \frac{\eta_d}{(\lambda+\alpha)^d} + \ldots$$

where $\alpha > 0$. If we use Padé-type approximation, then we multiply both sides by λ and require that the expansions in terms of $(\lambda+\alpha)^j$ agree up to the highest possible degree. A little calculation yields the approximation

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

This can alternatively be obtained by considering the following expansion

$$\begin{aligned}
\sum_{i=1}^{d} \frac{\alpha^{i-1}}{(\lambda+\alpha)^i} &= \frac{1}{\lambda+\alpha} \sum_{i=0}^{d-1} \left(\frac{\alpha}{\lambda+\alpha}\right)^i \\
&= \frac{1}{\lambda+\alpha} \times \frac{1-\left(\frac{\alpha}{\lambda+\alpha}\right)^d}{1-\frac{\alpha}{\lambda+\alpha}} \\
&= \frac{1}{\lambda}\left[1-\left(\frac{\alpha}{\lambda+\alpha}\right)^d\right].
\end{aligned} \tag{2}$$

The relative error made, which is

$$e(\lambda) = \left(\frac{\alpha}{\alpha+\lambda}\right)^d, \tag{3}$$

is illustrated in Figure 1 for eight different values of α, with $d = 3$, using values of α ranging from 0.025 to $\alpha = 0.375$ with increments of 0.05. Notice that for large λ all curves are fairly accurate. For the smaller values of λ, the quality of the approximation is still excellent and stays close to zero for small α. Thus, a good approximation to A^{-1} is given by

$$A^{-1} \simeq \sum_{i=1}^{d} \alpha^{i-1}(A+\alpha I)^{-i}. \tag{4}$$

Another approach leading to the same rational approximation is to consider the solution $x(z)$ to the system

$$(A+zI)x(z) = b.$$

The Taylor expansion at the point α of the solution $x(z)$ is given by

$$x(z) = \sum_{k\geq 0} x_k (z-\alpha)^k, \tag{5}$$

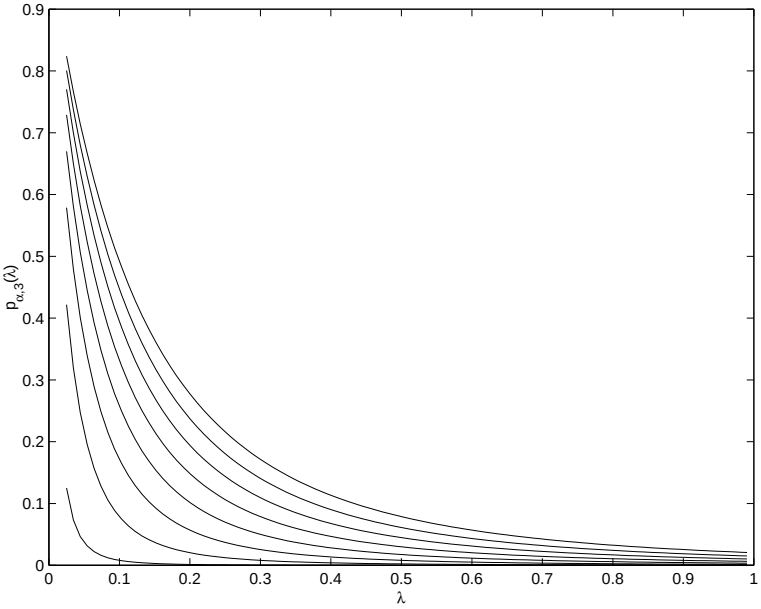

Figure 1: Relative error in interval $[0.025, 1]$.

where the vectors x_k are solutions to the systems

$$\begin{aligned} (A+\alpha I)x_0 &= b, \\ (A+\alpha I)x_{k+1} &= -x_k, \quad k \geq 0 \ . \end{aligned}$$

The above relations are obtained by identifying the coefficients of $(z-\alpha)^k$ in

$$(A + \alpha I + (z-\alpha)I) \sum x_k (z-\alpha)^k = b.$$

If the origin belongs to the disk of convergence of the series (5), then it is in theory possible to obtain an approximation to the solution $x = x(0)$ by evaluating the truncated series

$$\begin{aligned} x(0) &\simeq \sum_{k=0}^{d-1} x_k (-\alpha)^k \\ &= \left[(A+\alpha I)^{-1} + \alpha (A+\alpha I)^{-2} + \cdots + \alpha^{d-1} (A+\alpha I)^{-d} \right] b, \end{aligned} \tag{6}$$

where the same matrix than in equation (4) can be recognized.

Such power series expansions are used in industrial applications for computing the solution to a problem which depends on some parameters [2, 6]. When the matrix A is ill-conditioned, even if it is symmetric positive definite, the direct computation of (6) gives poor results because 0 is almost on the boundary of the disk of convergence: the convergence radius of (5) is exactly $\alpha + \lambda_1$ where $\lambda_1 > 0$ is the smallest eigenvalue of A. A Padé approximation could

be used instead of a Taylor polynomial, but it is still not very efficient. However, the matrix involved in (6) can be considered a good candidate for preconditioning the original system.

The attraction of the above expansion is that it allows to 'extrapolate' an approximate LU factorization of a close-by matrix to provide good convergence. Consider for example an ILUT factorization [11] of the matrix with a small α

$$A + \alpha I = LU + R, \quad M_\alpha \equiv LU. \tag{7}$$

Then the preconditioning operation used for preconditioning the original matrix A is defined by

$$M^{-1}v = \sum_{i=1}^{d} \alpha^{i-1} M_\alpha^{-i} v. \tag{8}$$

In many of our tests the degree d was between 1 and 10, while α could be taken to be between 10^{-4} and 1 (for a matrix whose rows are scaled by their norms). Assuming that M_α is the incomplete LU factorization for $A + \alpha I$ defined above, the algorithm for computing $w = M^{-1}v$ is as follows.

ALGORITHM **2.1** *Rational Preconditioner Operation.*

1. $w := v$
2. *Do* $j = 1 : d - 1$
3. $\qquad w := v + \alpha M_\alpha^{-1} w$
4. *EndDo*
5. $w := M_\alpha^{-1} w$

2.2 Inner-outer rational preconditioning

There is an interesting alternative to the above approach whose goal is to extract an optimal solution from the iterates of Algorithm 2.1. Instead of a preconditioning of the form (8) we seek a preconditioned vector of the form:

$$M^{-1}v = \sum_{i=1}^{d} \alpha_i M_\alpha^{-i} v,$$

where the scalars α_i are determined to minimize the residual norm $\|Aw - v\|_2$. An Arnoldi-Krylov basis can be generated for the preconditioned Krylov subspace

$$K_{\alpha,d} = \text{span}\{M_\alpha^{-1}Av, \ldots, (M_\alpha^{-1}A)^d v\},$$

and the usual least-squares solution obtained by GMRES is calculated to minimize the residual norm. This leads to an inner-outer iteration. In that case, the preconditioner is modified at each step, hence some flexible outer iteration algorithm like FGMRES [12] must be used. Numerical results indicate that this is more costly but often more effective than the simple expansion (8).

2.3 Shifting with an arbitrary diagonal

It is sometimes more attractive to be able to compute the ILU factorization of a matrix of the form

$$B_\alpha = A + \alpha D,$$

where D is a certain diagonal. In many instances, the matrix A is diagonally dominant in most of its rows but may have a few strongly non-diagonally dominant rows. It is then natural to shift only these rows, which means that the desired perturbation of the matrix is of the form $B_\alpha = A + \alpha D$. Note that the scalar α could be removed, but it is left here only for generality.

It is easy to extract a similar rational preconditioner for this case. A simple way to achieve this is to simply use the above idea for the matrix AD^{-1}, and then deduce the correct expansion. For the sake of completeness we show here the detail of the development. Let

$$C = (A + \alpha D)(\alpha D)^{-1}.$$

The expansion that is used is

$$\begin{aligned} C^{-1} + C^{-2} + \cdots + C^{-d} &= (I - C^{-1})^{-1} C^{-1} - (I - C^{-1})^{-1} C^{-d-1} \\ &= (C - I)^{-1} - (C - I)^{-1} C^{-d} \\ &= (\alpha D) A^{-1} - (\alpha D) A^{-1} C^{-d}. \end{aligned}$$

As a result,

$$A^{-1} = (\alpha D)^{-1} \left[C^{-1} + C^{-2} + \cdots + C^{-d} \right] + A^{-1} C^{-d}. \tag{9}$$

The scalar α can be integrated as part of a scaling of the matrix D , so we can set $\alpha = 1$. With this we obtain the approximation

$$A^{-1} \simeq D^{-1} \left[C^{-1} + C^{-2} + \cdots + C^{-d} \right].$$

Assuming that M_D is an approximate LU factorization of $A + D$ then, the equivalent of Algorithm 2.1 would be as follows.

ALGORITHM **2.2** *Rational Preconditioner Operation for Diagonal Shift.*

1. $w := v$
2. *Do* $j = 1 : d - 1$
3. $\qquad w := v + D M_D^{-1} w$
4. *EndDo*
5. $w := M_D^{-1} w$

2.4 Compounding ILU and shifting

We can take on further step in generalizing the principle of rational approximation by observing that in Algorithm 2.2, the matrix D is used only to perform a product with a vector and is never inverted. This suggests generalizing the above ideas to use an arbitrary matrix D which

is not restricted to being a diagonal. In fact, the idea that comes to mind immediately is to use for D the exact difference between A and an approximate LU factorization M_Δ in which diagonal shifting is used to prevent instability. The idea resembles that of a multiple-step SSOR or ILU, in which several steps of ILU or SSOR preconditioning operations are taken at each GMRES iteration, instead of just one step. If the matrix D is set equal to $M - A$ in Algorithm 2.2 then we would obtain a simplification in Line 3 and the following algorithm results.

ALGORITHM **2.3** *Compound Perturbed Preconditioning Operation.*

1. $w := v$
2. *Do* $j = 1 : d-1$
3. $\qquad w := v + (I - AM_\Delta^{-1})w$
4. *EndDo*
5. $w := M_\Delta^{-1} w$

In the above algorithm, M_Δ is an LU factorization of an arbitrarily perturbed matrix A. In particular, we can perturb the diagonals of A as the ILU process progresses to prevent instability - and we can drop small elements as is usually done. For example we can obtain an ILU factorization of the matrix:

$$A + \alpha I = L_\alpha U_\alpha + R$$

and use the resulting factor $M_\alpha = L_\alpha U_\alpha$ as the matrix M_Δ in the above algorithm. In this case, we have $I - AM_\alpha^{-1} = (\alpha I - R)M_\alpha^{-1}$, and the preconditioning matrix of algorithm 2.3 corresponds exactly to

$$\begin{aligned} M_\alpha^{-1} \sum_{i=0}^{d-1} (I - AM_\alpha^{-1})^i &= M_\alpha^{-1}(AM_\alpha^{-1})^{-1}(I - [I - AM_\alpha^{-1}]^d) \\ &= A^{-1}\left(I - [(\alpha I - R)M_\alpha^{-1}]^d\right), \end{aligned} \tag{10}$$

whereas the preconditioning matrix of algorithm 2.1 was

$$\begin{aligned} \sum_{i=1}^{d} \alpha^{i-1} M_\alpha^{-i} &= M_\alpha^{-1}(I - \alpha M_\alpha^{-1})^{-1}(I - \alpha^d M_\alpha^{-d}) \\ &= (A - R)^{-1}\left(I - \alpha^d M_\alpha^{-d}\right). \end{aligned} \tag{11}$$

Of course, both preconditioners coincide when the factorization is exact, i.e., when $R = 0$. When $R \neq 0$ is small, say of order α, then we have a perturbation of A^{-1} of order α^d in (10), whereas the perturbation is only of order α in (11). Hence algorithm 2.3 seems more attractive than algorithm 2.1, the price being an extra matrix-vector multiplication.

Alternatively, preconditioner (10) can be obtained by considering the Taylor expansion of the function $x(z)$ defined by

$$\begin{aligned} M(z)x(z) &= b, \\ A + zI &= M(z) + R(z), \end{aligned}$$

where $M(\alpha) = M_\alpha$ and $R(\alpha) = A + \alpha I - M_\alpha$. In order to have $x(0) = A^{-1}b$, we require $M(0) = A$, and the simplest form for $R(z)$ is then $R(z) = zR_1$. We can put $M(z)x(z) = (A + z(I - R_1))x(z) = b$ in the form

$$(M_\alpha + (z-\alpha)(I - R_1)) \sum_{k\geq 0} x_k (z-\alpha)^k = b.$$

Here again, by identification of the coefficients of $(z-\alpha)^k$, the vectors x_k are found to be solution to the systems

$$\begin{aligned} M_\alpha x_0 &= b, \\ M_\alpha x_k &= -(I - R_1)x_{k-1}, \quad k \geq 1 \ , \end{aligned}$$

which yields

$$x_k = M_\alpha^{-1} \left[-(I - R_1)M_\alpha^{-1}\right]^k b,$$

and the Taylor approximation of order $d-1$ of $x(0) = A^{-1}b$ reads

$$\begin{aligned} x(0) &\simeq \sum_{k=0}^{d-1} x_k (-\alpha)^k \\ &= M_\alpha^{-1} \sum_{k=0}^{d-1} \left[\alpha(I - R_1)M_\alpha^{-1}\right]^k b \\ &= M_\alpha^{-1} \sum_{k=0}^{d-1} \left[I - AM_\alpha^{-1}\right]^k b, \end{aligned}$$

where the preconditioning operation (10) can be recognized.

Instead of $R(z) = zR_1$, more general expressions of $R(z)$ could be chosen of the form

$$R(z) = \sum_{k\geq 0} R_k z^k,$$

the only constraints to be satisfied by $R(z)$ being $R(0) = 0$ and $R(\alpha) = A + \alpha I - M_\alpha$. At this stage, a natural question is how can the coefficients R_k be selected?

2.5 A multiscale-type procedure using different shifts

It is often observed that after a certain number of steps the convergence of GMRES slows down considerably, sometimes to the point of stagnating. This usually means that certain modes are not captured by the iterative process. Assume that the incomplete factorization is exact and consider

$$A + \alpha I = L_\alpha U_\alpha.$$

According to (11) with $R = 0$, the preconditioning matrix which is defined by Algorithm 2.1 is equal to

$$M^{-1} = A^{-1}\left[I - \alpha^d (A + \alpha I)^{-d}\right],$$

and so the residual matrix, which is,

$$I - AM^{-1} = \alpha^d (A + \alpha I)^{-d},$$

has eigenvalues:

$$\rho_i = \left(\frac{\alpha}{\lambda_i + \alpha} \right)^d ,$$

where λ_i is an arbitrary eigenvalue of A. These are the same functions as the errors in (3) and they are shown in Figure 1. It is clear that for large α those residual components associated with eigenvalues λ_i that are close to zero will not be reduced much. Notice that

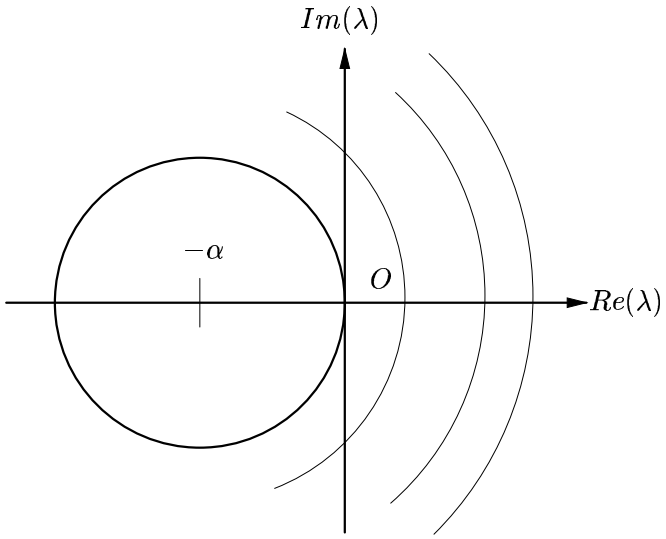

Figure 2: Damping region for preconditioner is outside the disk

any eigenvalue outside the disk $\overline{D}(-\alpha, \alpha)$ will be damped, i.e., it will be transformed into an eigenvalue smaller than one. Eigenvalues inside the disk can cause serious difficulties since they can be amplified and become very large if d is large.

Assuming that there are no eigenvalues inside the disk (as is the case for positive definite matrices), all damping ratios will be less than one. The farther away is λ_i from the center $-\alpha$ the smaller will be the damping ratio. Those eigenvalues close to the circle of center $-\alpha$ and radius α will have a damping ratio close to one. The concentric arcs in Figure 2 show the lines where the eigenvalues have the same damping factor ρ. If α is not changed during the iteration process, then the eigen-components of the residual which have small damping ratio (corresponding to large eigenvalues) will be eliminated quickly. Those with damping ratios close to one (for example those close to the origin) may remain little changed. What would be ideal is to have a procedure that does not disturb those small residual components achieved in earlier steps - but that reduces those closer to the origin further. This can be easily done by reducing α, say, at the occasion of a restart in the GMRES(m) algorithm. Experiments do indeed show that this principle works: similar to other multiscale methods,

it is much better to work on different parts of the spectrum - using different stages - rather than using a preconditioner based on a single α.

2.6 A multiple-pole rational expansion

It is possible to generalize the expansion (1) to one which uses several poles $-\alpha_i$. The most general expansion is of the form

$$s(\lambda) \simeq \frac{\beta_1}{\lambda+\alpha_1} + \frac{\beta_2}{(\lambda+\alpha_1)(\lambda+\alpha_2)} + \cdots \frac{\beta_d}{(\lambda+\alpha_1)\cdots(\lambda+\alpha_d)}.$$

The advantage of using the above expansion is that we can better exploit the use of different shifts as was motivated in the previous section. The coefficients β_i are easily determined as is shown next.

Proposition 2.1 *Let α_i be d arbitrary nonzero numbers and define for any $\lambda \neq -\alpha_1, \ldots, -\alpha_d$, and for $i = 1, \ldots, d$:*

$$\pi_i(\lambda) = \frac{1}{\alpha_i} \prod_{j=1}^{i} \left(\frac{\alpha_j}{\lambda+\alpha_j} \right), \tag{12}$$

$$\sigma_i(\lambda) = \sum_{j=1}^{i} \pi_j(\lambda). \tag{13}$$

Then the following equality holds

$$\frac{1}{\lambda} = \sigma_d(\lambda) + \alpha_d \frac{\pi_d(\lambda)}{\lambda}. \tag{14}$$

Proof. The proof is by induction and is straightforward. ∎

The above equality provides an approximation $\sigma_d(\lambda)$ to $s(\lambda)$ which is a natural generalization of expression (1).

The relative error (3) is replaced by

$$e(\lambda) = \prod_{j=1}^{d} \left(\frac{\alpha_j}{\alpha_j+\lambda} \right) \tag{15}$$

It is interesting to note that, surprisingly, the error does not depend on the order in which the poles are taken.

The corresponding modification of Algorithm 2.1 is quite simple. Since the order of the shifts has no effect, we can simply replace M_α in Line 3 by M_{α_j} and the M_α in Line 5 by M_{α_d}.

3 Analysis

In this section we give some error bounds for the CG algorithm applied to the solution of the preconditioned system

$$Bx := M^{-1}Ax = M^{-1}b. \tag{16}$$

Here we assume that the LU factorization is exact, i.e., $R = 0$ in (7). The rational transformation $B = r(A)$ makes the spectrum of B clustered around one, and the usual error bound for the CG algorithm can be improved, both for the first iterations and asymptotically.

The matrix A is supposed to be symmetric positive definite, with increasingly ordered eigenvalues λ_i,

$$0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n = 1,$$

associated to normalized eigenvectors $v_i, i = 1, \cdots, n$.

The preconditioning operation corresponding to (2) is modified by a constant β into

$$r(\lambda) = \frac{1}{\beta}\left[1 - \left(\frac{\alpha}{\lambda + \alpha}\right)^d\right], \quad \beta = 1 - \left(\frac{\alpha}{1+\alpha}\right)^d, \tag{17}$$

where $r(0) = 0$, $r(1) = 1$, and the eigenvalues of $B := r(A)$ are

$$\mu_i = r(\lambda_i), \quad 1 \leq i \leq n. \tag{18}$$

The choice of β ensures that $\mu_n = r(\lambda_n) = 1$. As the matrix A is supposed to be ill-conditioned, the eigenvalue λ_1 is very close to 0, and the shift α can be chosen greater than λ_1, but still noticeably smaller than 1. Hence $\beta \simeq 1$. The matrix B remains symmetric positive definite, and has the same eigenvectors v_i as A.

Let $x = A^{-1}b$ be the exact solution. We denote by x_m the approximate solution obtained at the m-th step of the CG algorithm applied to the matrix A, and by y_m the approximate solution obtained with the matrix B, starting with $y_0 = x_0$. The error on y_m is given by

$$\|y_m - x\|_B^2 = (B(y_m - x),\, y_m - x),$$

and a well-known optimality property which is satisfied by the iterates of the CG algorithm is that

$$\|y_m - x\|_B^2 = \min_{p \in \mathbb{P}_m,\ p(0)=1} \|p(B)(x_0 - x)\|_B^2, \tag{19}$$

where $\mathbb{P}_m$ is the set of polynomials of degree at most m. The minimizer p_m of (19) and the vector y_m are related by $y_m - x = p_m(B)(x_0 - x)$.

The classical error bound for the CG algorithm applied to the solution of the system $Ax = b$ is given by

$$\|x_m - x\|_A^2 \leq 4\left(\frac{1 - \sqrt{\lambda_1}}{1 + \sqrt{\lambda_1}}\right)^{2m} \|x_0 - x\|_A^2, \tag{20}$$

whereas the error for the preconditioned system (16) is

$$\|y_m - x\|_B^2 \leq 4\left(\frac{1 - \sqrt{\mu_1}}{1 + \sqrt{\mu_1}}\right)^{2m} \|x_0 - x\|_B^2. \tag{21}$$

3.1 Error bounds for the first iterations

For given $k \geq 0$ and $m \geq 1$, let

$$p(t) = \frac{T_m\left[1 + 2\frac{\mu_{k+1}-t}{1-\mu_{k+1}}\right]}{T_m\left[1 + 2\frac{\mu_{k+1}}{1-\mu_{k+1}}\right]}, \tag{22}$$

be the scaled Chebyshev polynomial that is small in the interval $[\mu_{k+1}, 1]$ (recall that $\mu_n = 1$). It satisfies $p(0) = 1$ and the following bounds, see e.g., [11],

$$p^2(t) \leq 1, \quad \forall t \in [0,1], \tag{23}$$

$$p^2(t) \leq \zeta := 4\left(\frac{1-\sqrt{\mu_{k+1}}}{1+\sqrt{\mu_{k+1}}}\right)^{2m}, \quad \forall t \in [\mu_{k+1}, 1]. \tag{24}$$

We have

$$\|p(B)(x_0 - x)\|_B^2 = \sum_{i=1}^{n} p^2(\mu_i) e_i^2 \mu_i, \quad e_i^2 \mu_i \geq 0,$$

where $e_i = (x_0 - x, v_i)$ is the initial error projected on the eigenvector v_i. The initial error $\|x_0 - x\|_B^2$ can be split into

$$\begin{aligned} \|x_0 - x\|_B^2 &= S_1 + S_2, \\ S_1 &= \sum_{i=1}^{k} e_i^2 \mu_i, \quad S_2 = \sum_{i=k+1}^{n} e_i^2 \mu_i, \end{aligned}$$

and weighting factors ρ_1 and ρ_2 can be chosen such that

$$\begin{cases} \rho_1 S_1 + \rho_2 S_2 = \|x_0 - x\|_B^2, \\ \rho_2 = \rho_1 \zeta. \end{cases}$$

Then, using (23) and (24), we obtain

$$\begin{aligned} \|p(B)(x_0 - x)\|_B^2 &= \sum_{i=1}^{k} p^2(\mu_i) e_i^2 \mu_i + \sum_{i=k+1}^{n} p^2(\mu_i) e_i^2 \mu_i \\ &\leq \sum_{i=1}^{k} e_i^2 \mu_i + \sum_{i=k+1}^{n} \zeta e_i^2 \mu_i = \frac{1}{\rho_1}(\rho_1 S_1 + \rho_2 S_2) = \frac{\|x_0 - x\|_B^2}{\rho_1}. \end{aligned}$$

Due to the definition of ρ_1, we have

$$\rho_1 = \frac{\|x_0 - x\|_B^2}{S_1 + \zeta S_2},$$

and it follows that

$$\|p(B)(x_0 - x)\|_B^2 \leq S_1 + \zeta S_2 \leq S_1 + \zeta \|x_0 - x\|_B^2.$$

Thus we obtain the error bound

$$\|y_m - x\|_B^2 \leq S_1 + 4\left(\frac{1-\sqrt{\mu_{k+1}}}{1+\sqrt{\mu_{k+1}}}\right)^{2m} \|x_0 - x\|_B^2. \tag{25}$$

One can observe that $S_1 = 0$ for the special case $k = 0$, and inequality (21) is then retrieved.

Up to now, we have not used the clustering of the spectrum of $B = r(A)$ around one. One consequence of this clustering is that $S_1 = S_1(k)$ may be small simultaneously with μ_{k+1} close to 1. We have $\mu_{k+1} = r(\lambda_{k+1})$ with

$$\lambda_{k+1} = c\alpha$$

for a certain positive number c. We keep in mind that we want to use an α larger than λ_1 but smaller than 1, which gives the possible magnitude of c. We choose also the smallest L, independent of α and k, for which

$$\sum_{i=1}^{k} e_i^2 \leq \lambda_k L \|x_0 - x\|^2, \quad \forall k = 1, 2, \ldots, n. \tag{26}$$

For example, if the matrix A has a uniform spectrum $\lambda_i = i/n$ and if $e_i^2 = 1$ for all i, then (26) holds for $L = 1$ (recall that $\|x_0 - x\|^2 = \sum_{i=1}^{n} e_i^2$). A large value of L would correspond to a clustering of the spectrum of A near 0, or to an initial error $x_0 - x$ essentially concentrated on the eigenspace associated to the smallest eigenvalues. Then we have the following error bound, which explains the fast decay of the error observed at the beginning of the iteration process.

Proposition 3.1 *For $\alpha > 0$, let $c > 0$ be chosen such that $c\alpha = \lambda_{k+1}$ is an eigenvalue λ_{k+1} of the matrix A. Then*

$$\|y_m - x\|_B^2 \leq \left[c\alpha L + 4\left(\frac{1}{\sqrt{(c+1)^d} + \sqrt{(c+1)^d - 1}}\right)^{4m}\right] \|x_0 - x\|^2. \tag{27}$$

Proof. Since $\mu_i \leq 1$ and by assumption (26), we have

$$S_1 = \sum_{i=1}^{k} e_i^2 \mu_i \leq \sum_{i=1}^{k} e_i^2 \leq \lambda_k L \|x_0 - x\|^2.$$

It follows from (25), $\lambda_k \leq \lambda_{k+1} = c\alpha$, $\|\cdot\|_B \leq \|\cdot\|$ and $\mu_{k+1} = r(\lambda_{k+1})$ that

$$\|y_m - x\|_B^2 \leq \left[c\alpha L + 4\left(\frac{1-\sqrt{r(c\alpha)}}{1+\sqrt{r(c\alpha)}}\right)^{2m}\right] \|x_0 - x\|^2.$$

For $0 < r < 1$, and since $\beta r < r$, we have

$$\frac{1-\sqrt{\ r}}{1+\sqrt{r}} \leq \frac{1-\sqrt{\beta r}}{1+\sqrt{\beta r}} = \left(\frac{\sqrt{1-\beta r}}{1+\sqrt{\beta r}}\right)^2.$$

then, using

$$\beta r(c\alpha) = 1 - 1/(c+1)^d,$$

we obtain

$$\begin{aligned} \frac{1-\sqrt{r(c\alpha)}}{1+\sqrt{r(c\alpha)}} &\leq \left(\frac{\sqrt{1/(c+1)^d}}{1+\sqrt{1-1/(c+1)^d}} \right)^2 \\ &= \left(\frac{1}{\sqrt{(c+1)^d}+\sqrt{(c+1)^d-1}} \right)^2. \end{aligned} \tag{28}$$

which completes the proof. ∎

We can notice that if we use the polynomial

$$p(t) = \frac{T_j\left[1+2\frac{\mu_{k+1}-t}{1-\mu_{k+1}}\right]}{T_j\left[1+2\frac{\mu_{k+1}}{1-\mu_{k+1}}\right]} \times \frac{T_{m-j}\left[1+2\frac{\mu_1-t}{1-\mu_1}\right]}{T_{m-j}\left[1+2\frac{\mu_1}{1-\mu_1}\right]},$$

instead of (22), then the following bound is obtained for $1 \leq j \leq m$:

$$\|y_m - x\|_B^2 \leq 4\left[c\alpha L + 4\left(\frac{1}{\sqrt{(c+1)^d}+\sqrt{(c+1)^d-1}}\right)^{4j}\right]\left(\frac{1-\sqrt{\mu_1}}{1+\sqrt{\mu_1}}\right)^{2(m-j)}\|x_0 - x\|^2.$$

However, for large m, we will see in the following section that a much stronger bound can be obtained.

3.2 Asymptotic error bounds

A strategy based on an idea described in [13], which takes advantage of the clustering of the spectrum of $B = r(A)$ around one, is now used for obtaining asymptotic error bounds. Instead of standard Chebyshev polynomials that are small in the interval $[\mu_1, \mu_n]$ we will use the following modified polynomial

$$C_m(t) = \prod_{i=1}^{k}\left(\frac{\mu_i - t}{\mu_i}\right) \times \frac{T_{m-k}\left[1+2\frac{\mu_{k+1}-t}{\mu_n-\mu_{k+1}}\right]}{T_{m-k}\left[1+2\frac{\mu_{k+1}}{\mu_n-\mu_{k+1}}\right]}$$

This consists of two parts. The first is a product term which takes the value zero for the first k smallest eigenvalues $\mu_1, \ldots, \mu_k$. The second is a standard scaled Chebyshev polynomial which is small in the interval $[\mu_{k+1}, \mu_n]$. Note that C_m is of degree m and that $C_m(0) = 1$. Since $C_m(\mu_i) = 0$ for $i = 1, \ldots, k$, the maximum of C_m on the spectrum of B is

$$\max_{\mu_j \in \Lambda(B)} |C_m(\mu_j))| \leq \max_{j=k+1,\ldots,n} \prod_{i=1}^{k}\left|\frac{\mu_i - \mu_j}{\mu_i}\right| \times \frac{1}{T_{m-k}\left[1+2\frac{\mu_{k+1}}{\mu_n-\mu_{k+1}}\right]} \tag{29}$$

Proposition 3.2 *For $\alpha > 0$, let $c > 0$ be chosen such that $c\alpha = \lambda_{k+1}$ is an eigenvalue λ_{k+1} of the matrix A. Then, for $m \geq k$, we have*

$$\|y_m - x\|_B \leq \frac{2\kappa^k(\alpha)}{\left[\sqrt{(c+1)^d} + \sqrt{(c+1)^d - 1}\right]^{2(m-k)}} \|x_0 - x\|_B. \tag{30}$$

where

$$\kappa(\alpha) = \frac{1 - \left(\frac{\lambda_1+\alpha}{1+\alpha}\right)^d}{\left(\frac{\lambda_1+\alpha}{\alpha}\right)^d - 1}. \tag{31}$$

Proof. Consider first the product term

$$\max_{j=k+1,\ldots,n} \prod_{i=1}^{k} \left|\frac{\mu_i - \mu_j}{\mu_i}\right| = \prod_{i=1}^{k} \left|\frac{\mu_i - \mu_n}{\mu_i}\right| \leq \left[\frac{\mu_n - \mu_1}{\mu_1}\right]^k$$

Recall that $\mu_n = 1$. We denote by $\kappa(\alpha)$ the term $(1-\mu_1)/\mu_1$ inside the brackets:

$$\begin{aligned}
\kappa(\alpha) &= \frac{1 - \frac{1}{\beta}\left(1 - \alpha^d/(\lambda_1+\alpha)^d\right)}{\frac{1}{\beta}\left(1 - \alpha^d/(\lambda_1+\alpha)^d\right)} \\
&= \frac{1 - \alpha^d/(1+\alpha)^d - 1 + \alpha^d/(\lambda_1+\alpha)^d}{1 - \alpha^d/(\lambda_1+\alpha)^d} \\
&= \frac{\alpha^d/(\lambda_1+\alpha)^d - \alpha^d/(1+\alpha)^d}{1 - \alpha^d/(\lambda_1+\alpha)^d}.
\end{aligned}$$

Multiplying numerator and denominator by $(\lambda_1+\alpha)^d/\alpha^d$ yields (31). Next, the second term is bounded by

$$\frac{1}{T_{m-k}\left[1 + 2\frac{\mu_{k+1}}{1-\mu_{k+1}}\right]} \leq 2\left(\frac{1-\sqrt{\mu_{k+1}}}{1+\sqrt{\mu_{k+1}}}\right)^{m-k},$$

and, using (28) once more, we obtain for $\mu_{k+1} = r(c\alpha)$

$$\frac{1}{T_{m-k}\left[1 + 2\frac{\mu_{k+1}}{\mu_n-\mu_{k+1}}\right]} \leq 2\left(\frac{1}{\sqrt{(c+1)^d} + \sqrt{(c+1)^d - 1}}\right)^{2(m-k)}$$

which completes the proof. ∎

A fair comparison would be to compare one step of the rational preconditioner with d steps of the standard preconditioned conjugate gradient applied with some accurate ILU preconditioner. This is because applying $r(A)$ uses d solves with the LU factorization – which is likely to dominate the cost. For these d steps the convergence factor as inferred from the standard bound is given by

$$\tilde{\rho} = \left(\frac{1-\sqrt{\lambda_1}}{1+\sqrt{\lambda_1}}\right)^d$$

which is to be compared with the asymptotic convergence factor,

$$\rho(c) \leq \left(\frac{1}{\sqrt{(c+1)^d} + \sqrt{(c+1)^d - 1}} \right)^2$$

The above asymptotic argument ignores the potentially large constant in the numerator of (30). However, it gives a rough comparison of the situation at the asymptotic regime where, for example, a large accuracy is required.

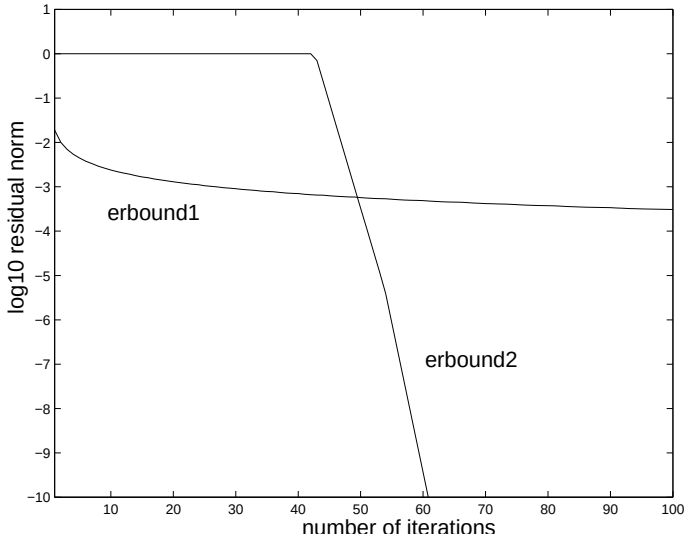

Figure 3: error bounds $\|y_m - x\|_B$.

The estimated error bounds resulting from Propositions 3.1 and 3.2 are illustrated by the two curves `erbound1` and `erbound2` on Figure 3. The following parameters were used: $d = 3$, $\lambda_1 = 10^{-15}$, $L = 10$, $n = 10^5$, $\alpha = 10^{-5}$ and the first eigenvalues were supposed to be

$$\lambda_k = \lambda_1 + \frac{k-1}{nL}.$$

3.3 Choice of the parameter α

Propositions 3.1 and 3.2 have shown that the rational preconditioned CG algorithm converges faster not only because of the shift on the smallest eigenvalue $\mu_1 > \lambda_1$, but also because of the clustering of the spectrum of the matrix B around one. Although one of the principal interest of the method is to allow a sparse incomplete LU factorization of $A + \alpha I$, the efficiency of Algorithm 2.1, when using this incomplete LU factorization, is difficult to analyze directly. However, some guidance for choosing the shift α can be obtained by supposing that $M_\alpha = A + \alpha I$ in Line 3 of Algorithm 2.1, and that each system $(A + \alpha I)u = v$ itself is solved by using the CG algorithm also. Then we can compare the computational cost of this method to the cost of the direct solution of the original system by the CG algorithm without preconditioning. Now the choice of α has to be a compromise between two conflicting demands:

- The smaller α is, the better the convergence of the rational preconditioned CG algorithm.
- However, a small α slows down the convergence of the CG algorithm for solving $(A + \alpha I)u = v$.

Let $\varepsilon > 0$ be the wanted error reduction $\|x_m - x\|/\|x_0 - x\|$ for the solution of $Ax = b$, and ε' the error reduction which is used for the solution of the intermediate systems $(A+\alpha I)u = v$. We denote respectively by N_A, N_B and N_α the estimated number of iterations for solving $Ax = b$ without preconditioning, the preconditioned system $M^{-1}Ax = M^{-1}b$, and each of the intermediate systems $(A + \alpha I)u = v$. Algorithm 2.1 requires dN_BN_α matrix-vector products, whereas the direct implementation of the CG algorithm requires N_A matrix-vector products. The question is which α minimizes dN_BN_α, and whether the ratio $\omega = dN_BN_\alpha/N_A$ is smaller than one.

The classical estimate for N_A follows from (20):

$$N_A = \frac{\log(2/\varepsilon)}{\log \rho(\lambda_1)}, \quad \rho(t) = \frac{1 + \sqrt{t}}{1 - \sqrt{t}}. \tag{32}$$

However this estimate does not take into account the rounding errors due to finite arithmetic.

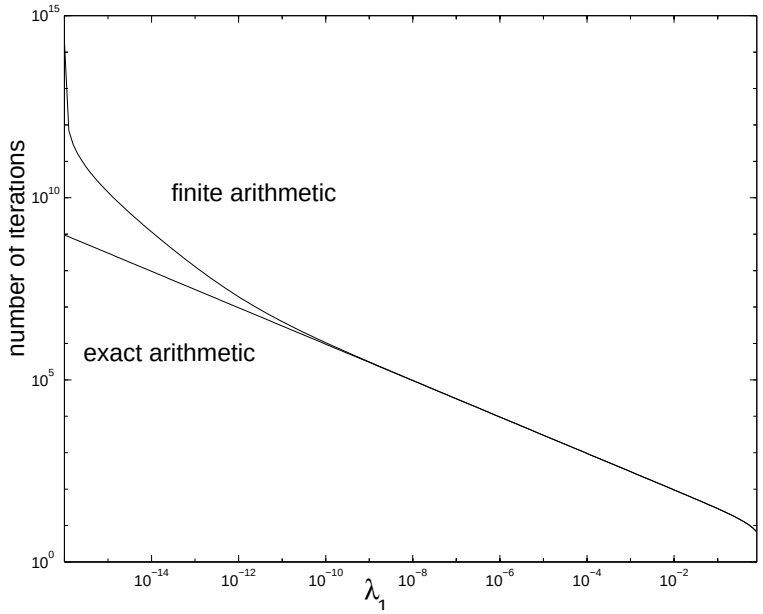

Figure 4: number of CG iterations.

For very small λ_1, say $0 < \lambda_1 \leq s$, the CG algorithm will in fact not converge at all, and $N_A = \infty$. For this reason we use the following estimate of N_A :

$$N_A \quad = \quad N(\varepsilon, \lambda_1), \tag{33}$$

$$N(\varepsilon,t) \;=\; \frac{\log(2/\varepsilon)}{\log\rho(t)} \times \left(1+\frac{\eta}{\frac{\log\rho(t)}{\log\rho(s)}-1}\right), \quad 0<s<t.$$

The two functions (32) and (33) are shown in Figure 4 for $s=10^{-16}$ and $\eta=100$. Using the same function for N_α we have

$$N_\alpha = N(\varepsilon', \lambda_1+\alpha).$$

Next we need to estimate N_B. For this purpose, we suppose here that $c=1$ in $\lambda_{k+1}=c\alpha$ (with $0\le k\le n-1$), and that

$$\alpha = \lambda_1 + \frac{k}{n-1}(1-\lambda_1),$$

which means that the spectrum of A is uniformly distributed between λ_1 and 1. Then it follows from Proposition 3.2 that

$$\|y_m - x\|_B \le \frac{2[\kappa(\alpha)\delta^2]^{k(\alpha)}}{\delta^{2m}}\|x_0-x\|_B,$$

with

$$\delta=\sqrt{2^d}+\sqrt{2^d-1},\;\; \kappa(\alpha)=\frac{1-\left(\frac{\lambda_1+\alpha}{1+\alpha}\right)^d}{\left(\frac{\lambda_1+\alpha}{\alpha}\right)^d-1},\;\; k(\alpha)=\frac{(n-1)(\alpha-\lambda_1)}{1-\lambda_1},$$

which yields

$$N_B=\frac{k(\alpha)\log(\kappa(\alpha)\delta^2)+\log(2/\varepsilon)}{2\log\delta}.$$

The ratios $\omega(\alpha)=dN_BN_\alpha/N_A$ are illustrated in Figure 5 for different values $\lambda_1=10^{-9}$, 10^{-11}, 10^{-13}, 10^{-15}. The following parameters were used: $d=3$, $\varepsilon=10^{-6}$, $\varepsilon'=10^{-10}$, $s=10^{-16}$ and $\eta=100$. These estimates are far from being optimal for the rational preconditioned algorithm, since we have not taken into account that when solving the intermediate systems $(A+\alpha I)w=v$, some good initial guess can be obtained from the solution obtained at the previous step, which can significantly speed-up global convergence.

It is interesting to observe that the optimal α is not very sensitive to the value of the smallest eigenvalue λ_1. We could not obtain a closed form for the α which minimizes the function $\omega(\alpha)$. However, for $0<\lambda_1\ll\alpha\ll 1$, a very good approximation to this function is given by

$$\begin{aligned}\tilde{w}(\alpha) &= C\,\frac{n\alpha\log(\alpha\delta^2/(d\lambda_1))+\log(2/\varepsilon)}{2\sqrt{\alpha}}\\ C &= \frac{d\log(2/\varepsilon')\log(\rho(\lambda_1))}{2\log(\delta)\log(2/\varepsilon)},\end{aligned}$$

which is obtained by using the following approximations:

$$\log(\rho(\lambda_1+\alpha))\simeq 2\sqrt{\alpha},\;\; N_\alpha\simeq\frac{\log(2/\varepsilon')}{2\sqrt{\alpha}},$$

$$k(\alpha)\simeq n\alpha,\;\; \kappa(\alpha)\simeq\frac{\alpha}{d\lambda_1},\;\; N_B\simeq\frac{n\alpha\log(\alpha\delta^2/(d\lambda_1))+\log(2/\varepsilon)}{2\log(\delta)}.$$

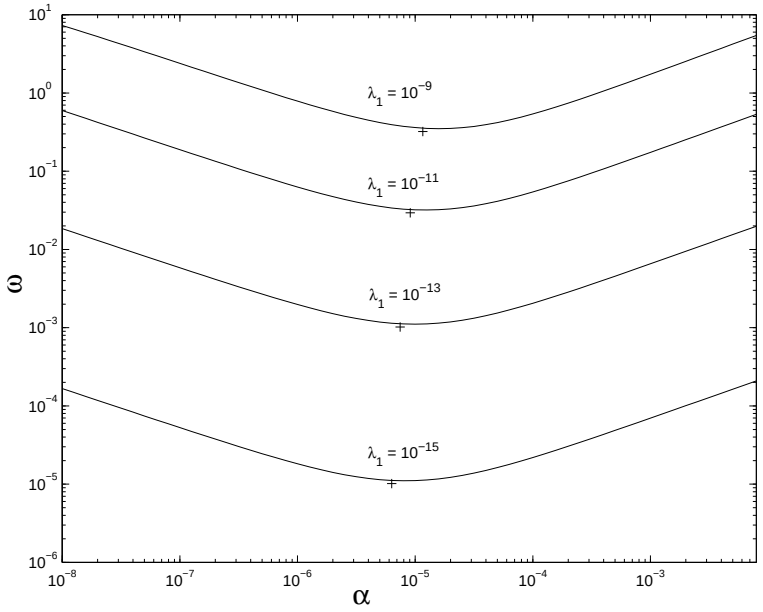

Figure 5: ratio $\omega = dN_B N_\alpha / N_A$

The minimum of $\tilde{\omega}(\alpha)$ is obtained for

$$\alpha \log\left(\frac{\alpha^2\delta^2}{d\lambda_1}\right) = \frac{1}{n}\log\left(\frac{2}{\varepsilon}\right).$$

Setting $s = {}^2\delta^2/(d\lambda_1)$ and $\gamma = \alpha s$, this equation reads

$$\gamma \log(\gamma) = \nu := \frac{s}{n}\log(2/\varepsilon)$$

and, for large ν, an approximate solution to this equation is given by

$$\gamma = \frac{\nu}{\log(\nu)},$$

which yields

$$\alpha = \frac{\nu}{s\log(\nu)} = \frac{\log(2/\varepsilon)}{n\log(\mathrm{e}^2\delta^2\log(2/\varepsilon)/(nd\lambda_1))}.$$

These approximate values $(\alpha, \tilde{\omega}(\alpha))$ are indicated with a plus sign in Figure 5. It can be observed that the minima are obtained with a sufficient accuracy.

The functions $n \times \alpha(\lambda_1)$ are illustrated in Figures 6 and 7 for different values of the dimension n and the error reduction ε (notice that $\alpha(\lambda_1)$ is independent of the inner error reduction ε'). The other parameters are $d = 3$, $\varepsilon = 10^{-8}$ for Figure 6 and $n = 10^5$ for Figure 7. A good candidate for α can be simply $1/n$.

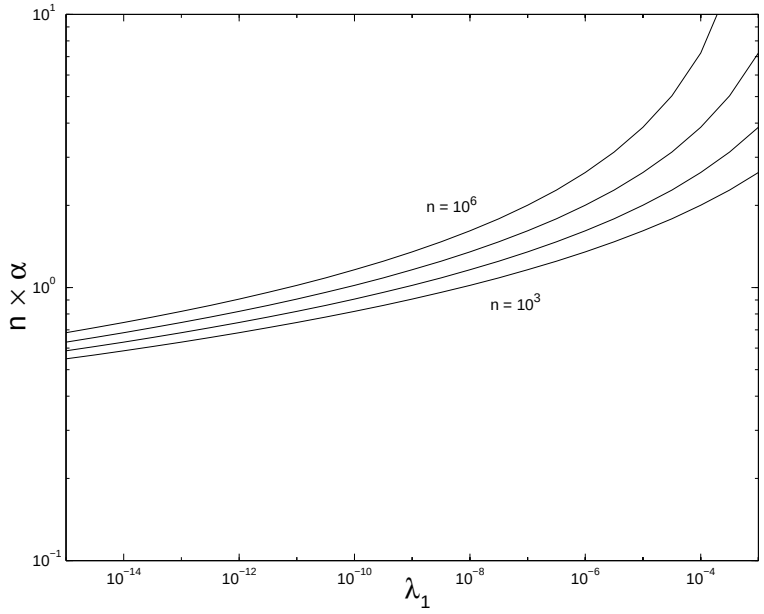

Figure 6: $n\times$(optimal α) versus λ_1 for $n = 10^3$, 10^4, 10^5, 10^6

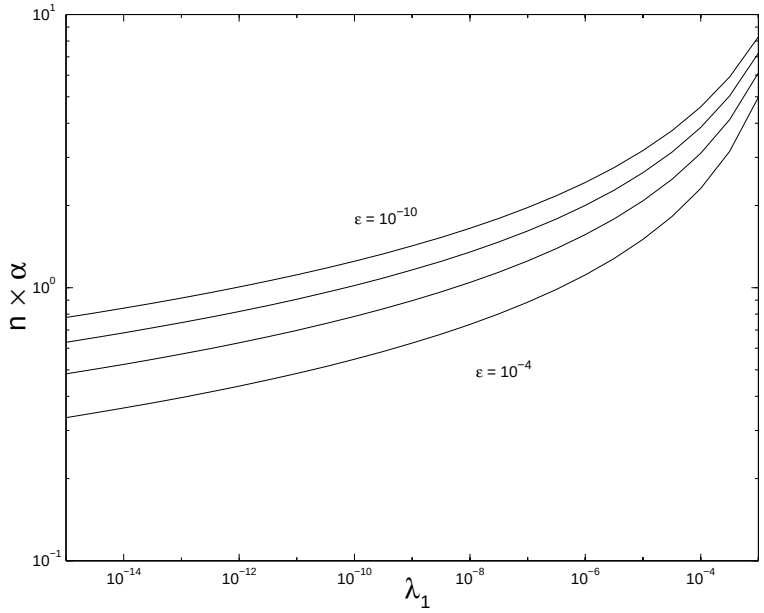

Figure 7: $n\times$ (optimal α) versus λ_1 for $\varepsilon = 10^{-4}$, 10^{-6}, 10^{-8}, 10^{-10}

Finally, for a general symmetric positive definite matrix with smallest eigenvalue λ_1 and largest eigenvalue λ_n, the recommended value of the shift would become

$$\alpha = \frac{\lambda_n \log(2/\varepsilon)}{n \log(\mathrm{e}^2 \delta^2 \log(2/\varepsilon) \lambda_n / (nd\lambda_1))},$$

and a good candidate for α would be simply λ_n/n. The value λ_n itself can be estimated by a few iterations of the power method.

4 Numerical results

This section reports on a few numerical experiments with the rational preconditioning techniques for solving difficult real-world problems in structural mechanics. For these problems, a straightforward application of standard preconditioning techniques, such as an incomplete LU factorization, fails due to their instability. Diagonal shift and large fill-in may be needed to achieve convergence as investigated in [15] for tire design problems. However, choosing the most appropriate shift value can be very time consuming, and the preconditioning becomes quite expensive to apply in the case of large fill-in. We attempt to show how rational preconditioning can handle these difficulties.

4.1 Test problems and the components of the iterative solution

For the experiments, we have selected a few linear systems arising in shell modeling and tire design. Table 1 gives some information about the problems. The matrix ELTCOQUE comes from the discretization of thin shells, using DKT12 elements (Discrete Kirchoff Triangle with 12 ddl, [1]), and was provided by CADOE S.A. The matrices MCHLNF and MCHLNE come from the discretization of nonlinear static equilibrium equations in tire problems, and were provided by MICHELIN. Columns denoted by n and n_z give the numbers of rows and nonzero entries in the matrices, respectively. Column `Dominance` shows the ratio of diagonally dominant rows to the matrix size. This number gives a good indication of the difficulty of the corresponding linear system. The matrices have a small percent of the diagonally dominant rows, and thus we can expect the linear systems to be quite difficult. All the matrices in Table 1 are *structurally* symmetric. The symmetry in value is indicated in Column `Symmetry`.

Table 1: Description of test problems.

`Name`	n	n_z	`Dominance`	`Symmetry`	`Matrix source`
ELTCOQUE	38002	949452	0.6%	Yes	Shell modeling
MCHLNF	49800	4136484	5%	Yes	Tire design
MCHLNE	49800	4136580	4.6%	No	Tire design

Restarted GMRES was used as the accelerator. In particular, its variant FGMRES(k) which allows variable preconditioning [11] was employed when preconditioner changes during

iteration. Deflated GMRES(k) [8], [3] was used in the cases of stagnation. In deflated GMRES(k) the eigenvectors corresponding to a few smallest eigenvalues are added to the Krylov subspace to prevent stalling of the GMRES(k) convergence. For both FGMRES(k) and deflated GMRES(k), the Krylov subspace dimension is equal to 54 and includes four injected eigenvectors in the case of deflated GMRES(k). We took a random initial guess with the right-hand side constructed such that the solution is the vector of all ones.

Rational acceleration is applied to the factorization produced by the Algebraic Recursive Multilevel Solver (ARMS) [14]. This choice of the preconditioner is motivated by the versatility of ARMS and its ability to solve efficiently the structural mechanics problems. ARMS is an algebraic multigrid-like algorithm that requires no underlying set of grids for defining prolongation and restriction operators. ARMS works by reordering the matrix in the block form

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix},$$

in which B is diagonal or block-diagonal with small blocks. The above matrix is then approximately block-factored as

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} \approx \begin{pmatrix} L & 0 \\ G & I \end{pmatrix} \begin{pmatrix} U & W \\ 0 & S \end{pmatrix}$$

using again dropping strategies. Then the reordering and factorization were repeated recursively on the Schur complement matrix S, for a small number of levels. At the last level the matrix S is factored using again a standard ILUT or ILUTP factorization. Both the construction of the preconditioner and the forward-backward solutions in ARMS are recursive. In addition ARMS allows inter-level iterations (referred to as W-cycles in the multigrid literature), though these tend to be fairly expensive if the number of levels is high.

A particular instance of the ARMS preconditioner as well as the ARMS performance for a given iterative algorithm are controlled by several parameters, such as the block size and number of levels specifying the block and level preconditioner structures, respectively. We allow no inner iterations in the levels of ARMS to reduce the time of the preconditioning operation. Varying the number of ARMS levels from 2 to 5 did not affect the preconditioner performance, but fewer levels make the preconditioner construction less expensive. Thus, the number of levels was chosen to be equal to two. Our experience shows that taking small blocks (of size 3 or 10) instead of larger blocks (of size 100) often yields better overall performance.

Filtering small (less the 10^{-3}) off-diagonal entries in the matrix from which the preconditioners are built speeds up their construction since fewer nonzero entries remain in the original and preconditioner matrices. We have observed that in the given problem types after such a filtering process, the majority of (weakly) diagonally dominant rows have *all* their off-diagonal entries dropped. The corresponding rows constitute an independent set, which we call the *trivial independent set*. These rows become properly permuted by setting to 1 the independent set parameter in ARMS.

4.2 Rational acceleration and the accuracy of preconditioning

With a rational acceleration (Algorithm 2.1), a less accurate preconditioning matrix (i.e., with a small fill-in) may be required to achieve a good convergence. For the MCHLNF and

MCHLNE problems, Table 2 shows the results of the three runs of an experiment in which the acceleration degree was increased in each run while the amount of preconditioner fill-in was halved. The total number of nonzero elements in the preconditioning matrix is shown in Column 2. Columns `Construction` and `Solution` give the preconditioner construction and solution times (in seconds), respectively. The number of *outer* deflated GMRES(54) iterations is stated in Column 5. The shift value α and the ARMS dropping tolerance have been kept constant and equal to 0.8 and 0.0, respectively. The reduction of 10^6 in the residual norm has been achieved by deflated GMRES(54).

Table 2: Dependence of the execution times on the degree of rational acceleration and the preconditioner accuracy.

Name	(Degree, Fill-in)	n_z	Construction	Solution	Iterations
MCHLNF	(2,240)	22,161,175	1327.12	1988.43	564
	(3,120)	11,301,437	726.81	1417.35	465
	(4,60)	5,710,647	410.15	1412.13	541
MCHLNE	(2,240)	22,161,330	1286.47	2256.35	626
	(3,120)	11,286,156	696.18	1575.95	508
	(4,60)	5,621,009	381.48	1406.52	550

As expected, the preconditioner construction time is almost proportional to the amount of fill-in and affects significantly the total execution time. The preconditioning application cost itself increases when the degree d of approximation grows, but decreases when the amount of fill-in is reduced. Similarly, increasing d reduces the number of outer iterations, while reducing the fill-in augments the number of outer iterations. These opposite tendencies finally result in a reduction of the solution time. Hence, increasing the degree and reducing the fill-in reduces both the construction and solution time, that is, in this example, the rational acceleration saves time and memory.

4.3 Shift and degree selection

Some guidelines for choosing the shift value α are proposed in subsection 3.3 for an exact factorization of $A + \alpha I$ used with the CG algorithm. Here, for a test problem, we show the dependences of the convergence rate and the stability of the preconditioner on α and outline an automatic process of arriving at an appropriate shift value. In [4], a strong correlation between stability of the preconditioner and the size of $\mathcal{E} = \log(\|(LU)^{-1}\|_{\inf})$ was shown and was suggested as a practical means of evaluating the quality of a preconditioner. We can inexpensively compute $\mathcal{E}_\alpha$ as

$$\mathcal{E}_\alpha = \log(\|(LU)^{-1}e\|_1),$$

where e is a vector of all ones and LU is the incomplete LU factorization of $A + \alpha I$.

For the problem ELTCOQUE, the amount of fill-in was equal to 30 and the dropping tolerance equal to 0 in the preconditioner construction. Without rational preconditioning

and without shift ($\alpha = 0$), there was no reduction of the residual norm. Figure 8 shows the convergence curves for different choices of α without rational approximation (i.e., its degree $d = 1$). When α is quite large (solid line) the convergence of flexible GMRES(54) is slow although the incomplete LU factors are stable ($\mathcal{E}_\alpha = 0.29$). It is possible to start with some large α (say, 0.8) and gradually decrease it as long as the indicator $\mathcal{E}_\alpha$ stays small. Changing α dynamically requires modifying the incomplete LU factors. Relatively inexpensive modifications could potentially be obtained by means of sparse approximate inverse techniques as mentioned in [5]. Devising an effective procedure for updating LU factors is beyond the scope of this paper. In the experiments, we refactor the shifted matrix A each time a new shift value is taken. Since this procedure is expensive, it is performed only at a GMRES restart. The dashed line in Figure 8 indicates that the iterative convergence is much faster for varying α than for some constant large α. The solution times are 222.22 and 303.54 seconds, respectively. Monitoring $\mathcal{E}_\alpha$ allows an early detection of a possible preconditioner instability for some small α indicating that it should not be decreased further. The dash-dotted and dotted curves show the convergence histories for the two constant shift values (3.8×10^{-2} and 6.25×10^{-3}, respectively) that precede sharp increases in the estimate of $\mathcal{E}_\alpha$ for two strategies of changing α. A more aggressive decrease, halving alpha at each restart (dotted line), quickly arrives at a shift value corresponding to an unstable preconditioner with the preceding value being already too small. A more gradual decrease in α allows to find the shift value more accurately.

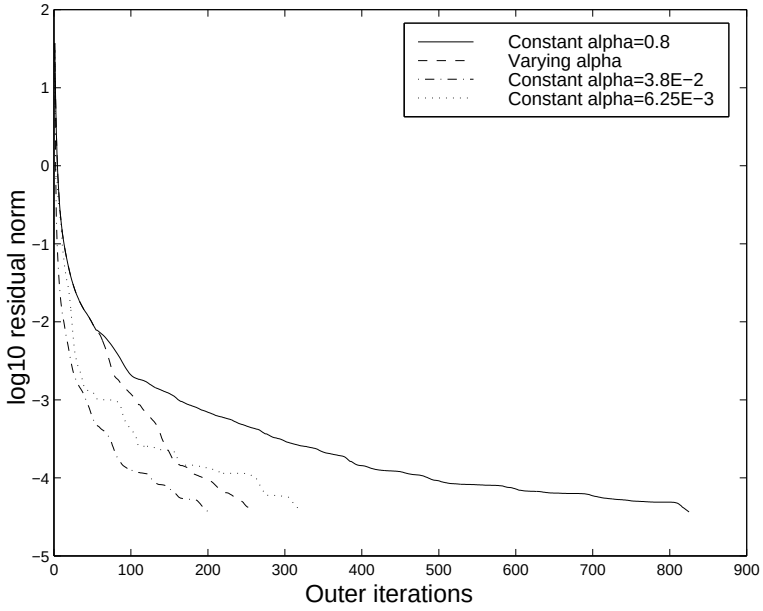

Figure 8: Choosing α for the problem ELTCOQUE without rational approximation.

The effect of an unstable preconditioning is especially pronounced when the shift value continues to be halved (dash-dotted curve in Figure 9), where the residual norm *increases* at about iteration 500. Figure 9 presents the convergence curves of the experiments in which an

iterative method attempts to achieve maximum accuracy in $1,000$ iterations given four ways to choose (α, d) in the rational approximation (Algorithm 2.3) of the preconditioning. Both the solid and dashed lines are for the case when alpha is decreasing slowly. The curve for a fixed degree of approximation is represented by the solid line. The case where the degree is increased by one at each restart, with initial degree 2, is represented by the dashed line. The dotted line corresponds to the constant smallest shift (6.25×10^{-3}) as given in Figure 8. Note that keeping α constant and small enough accelerates convergence in the first iterations, but the ultimate residual norm reduction may be much less than when the degree is varied and a large shift value is taken (dashed curve). Thus varying the degree as well as the shift α can be beneficial in achieving high accuracy in spite of an increase in the cost of the preconditioning operation.

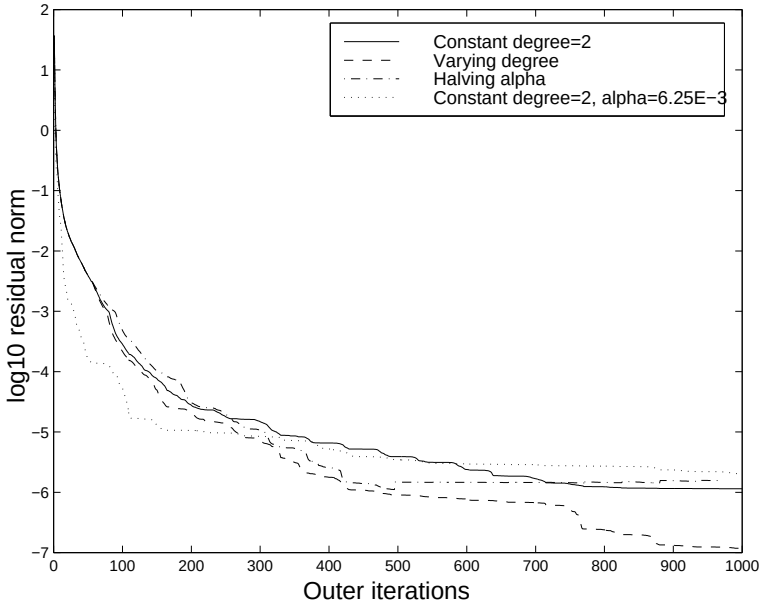

Figure 9: Achieving high accuracy for the problem ELTCOQUE with rational approximation.

5 Conclusion

We have shown a strategy for building an effective preconditioner for dealing with highly ill-conditioned matrices. The main difficulty with such matrices is that the standard ILU preconditioners tend to produce an ILU factorization that is often unstable. Instability is often avoided by using a very high level of fill-in to obtain an LU factorization that is very close to that of A. This approach may not be feasible because of its high memory and computational cost. The alternative proposed in this paper, is to shift the matrix before computing its ILU factorization, and then to use a rational expansion in order to increase the accuracy by

extrapolating it to approximate A^{-1}. We have explained why changing the shift or the degree during iteration helps refocus the iterative process in reducing residual components on different parts of the spectrum and can be quite important in achieving convergence. Numerical experiments support this hypothesis. They also show that the method can succeed in solving rather difficult problems without requiring an excessive amount of memory.

References

[1] J. L. Batoz, G. Dhatt, *Modélisation des structures par éléments finis, volume 3 : coques*, Hermès, Paris, 1992.

[2] J. D. Beley, C. Broudiscou, Ph. Guillaume, M. Masmoudi, F. Thevenon, *Application de la méthode des dérivées d'ordre élevé à l'optimisation des structures,* Revue Européenne des éléments finis, 5 (1996), 537-567.

[3] A. Chapman and Y. Saad, *Deflated and augmented Krylov subspace techniques*, Numerical Linear Algebra with Applications, 4 (1997), pp. 43–66.

[4] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, Journal of Computational and Applied Mathematics, 87 (1997), pp. 387–414.

[5] ——, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM Journal on Scientific Computing, 19 (1998), pp. 995–1023.

[6] Ph. Guillaume and M. Masmoudi, *Solution to the time-harmonic Maxwell's equations in a waveguide, use of higher order derivatives for solving the discrete problem*, SIAM J. on Num. Anal. 34-4, (1997), 1306-1330.

[7] T. A. Manteuffel, *An incomplete Factorization technique for positive definite linear systems,* Math. of Comp., 34 (1980), 473-497.

[8] R. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., Vol. 16 (1995), pp. pages 1154–1171.

[9] B. N. Parlett, *The Symmetric Eigenvalue Problem,* Prentice Hall, Englewood Cliffs, 1980.

[10] A. Ruhe, *Rational Krylov algorithms for eigenvalue computation,* Linear Algebra Appl., 58 (1984), 391-405.

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

[12] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm,* SIAM J. Sci. Comput. 14 (1993), no. 2, 461–469.

[13] Y. Saad, *Theoretical Error Bounds and General Analysis of a few Lanczos-Type Algorithms,* in Proceedings of the Cornelius Lanczos International Centenary Conference, Editors J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, SIAM, 1994, 123-134.

[14] Y. Saad and B. Suchomel. ARMS: An algebraic recursive multilevel solver for general sparse linear systems. Technical Report umsi-99-107, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1999.

[15] M. Sosonkina, J. Melson, and L. Watson, *Iterative Solution of Large Linear Systems Arising in Tire Design*, in Modeling and Simulation Based Engineering, S. Atluri and P. Donoghue, eds., vol. 1, Tech Science Press, 1998, pp. 473–478.