CAM 4201
pp: 1-24 (col.fig.: Nil)
PROD. TYPE: COM
ED: Indira
PAGN: VD – SCAN: Mouli

ARTICLE IN PRESS

ELSEVIER

Available at WWW.MATHEMATICSWEB.ORG POWERED BY SCIENCE DIRECT®

Journal of Computational and Applied Mathematics ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

Rational approximation preconditioners for sparse linear systems

Philippe Guillaume[a], Yousef Saad[b,1], Masha Sosonkina[c,*,2]

[a] *UMR MIP 5640, Département de Mathématiques, INSA, Complexe Scientifique de Rangueil, 31077 Toulouse Cedex, France*

[b] *Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455, USA*

[c] *Department of Computer Science, University of Minnesota-Duluth, 320 Heller Hall, 1114 Kirby Drive, Duluth, MN 55812-2496, USA*

Received 12 March 2002; received in revised form 3 February 2003; accepted 18 March 2003

Abstract

This paper presents a class of preconditioning techniques which exploit rational function approximations to the inverse of 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 for 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 nonshifted 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.

Keywords: ■; ■; ■

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

* Corresponding author.
E-mail addresses: guillaum@gmm.insa-tlse.fr (P. Guillaume), saad@cs.umn.edu (Y. Saad), masha@d.umn.edu (M. Sosonkina).

[1] This work was supported in part by the U.S. Army Research Office under grant DAAD19-00-1-0485, and in part by the Minnesota Supercomputer Institute.

[2] This work was supported by Michelin Americas Research and Development Corporation.

0377-0427/03/$ - see front matter © 2003 Published by Elsevier Science B.V.
doi:10.1016/S0377-0427(03)00480-1

be very difficult to solve by iterative methods despite the fact that they are symmetric positive definite. One source of difficulty 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., [8]. This by itself may not be sufficient.

A modification of this idea leads 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 rather accurate representation of $A+\alpha I$. It is also more likely to be stable, in the sense defined in [6], in that its inverse does not have an extremely large norm. Instability of preconditioners in this sense is often the main cause of difficulty with incomplete LU factorization preconditioners of very ill-conditioned matrices. Thus, we can still manage to solve extremely ill-conditioned problems by exploiting incomplete factorizations of $A+\alpha I$, whereas an incomplete factorization for A would almost certainly result in failure.

Section 2 presents two algorithms for computing a rational preconditioner, which are illustrated on a simple example. Section 3 reports some error bounds for the conjugate gradient algorithm in the special case where the LU factorization of $A+\alpha I$ is computed exactly. 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, even when using an incomplete factorization. A second error bound shows that the rational transformation improves the rate of convergence at the asymptotic regime, where, for example, a large accuracy is required. Finally, some numerical experiments on solving difficult real-world problems in structural mechanics are reported in Section 4.

2. Rational approximation preconditioning

Consider the linear system

$$Ax = b,$$

where A is a nonsingular 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, preconditioners based on a polynomial approximation $p(A)$ of A^{-1} 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 [10], 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 [11] for eigenvalue computations. 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

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

where $\alpha > 0$. We can use a Padé-type approximation to match both sides of (1), in the variable $t \equiv \lambda + \alpha$ to the approximation: we multiply both sides by t^d and require that the expansions in terms of t^j agree on both sides up to the highest possible degree. A little calculation yields the approximation

$$s(\lambda) \simeq \sum_{i=1}^{d} \frac{\alpha^{i-1}}{(\lambda+\alpha)^i}. \tag{2}$$

This can alternatively be verified by considering the following expansion:

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

$$= \frac{1}{\lambda}\left[1 - \left(\frac{\alpha}{\lambda+\alpha}\right)^d\right]. \tag{3}$$

The relative error is given by

$$e(\lambda) = \left(\frac{\alpha}{\alpha+\lambda}\right)^d. \tag{4}$$

The function $\lambda s(\lambda)$ which gives an idea of the eigenvalues of the preconditioned matrix is illustrated in Fig. 1 for different values of α, with $d=3$ (left), and $d=6$ (right), using values of α ranging from 0.025 to $\alpha = 0.375$ with increments of 0.05. The smaller α is, the closer the curve to the constant one. Thus, for the left plot, the highest curve (close to one) corresponds to $\alpha=0.025$, and the lowest one to $\alpha = 0.025 + 7 \times 0.05 = 0.375$. For the right plot ($d=6$) only the lowest 4 curves are shown. 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 one for small α.

2.2. Compounding ILU and shifting

Substitution of A for λ in (3) leads to an approximation to A^{-1}, given by

$$A^{-1} \simeq \sum_{i=1}^{d} \alpha^{i-1}(A+\alpha I)^{-i}. \tag{5}$$

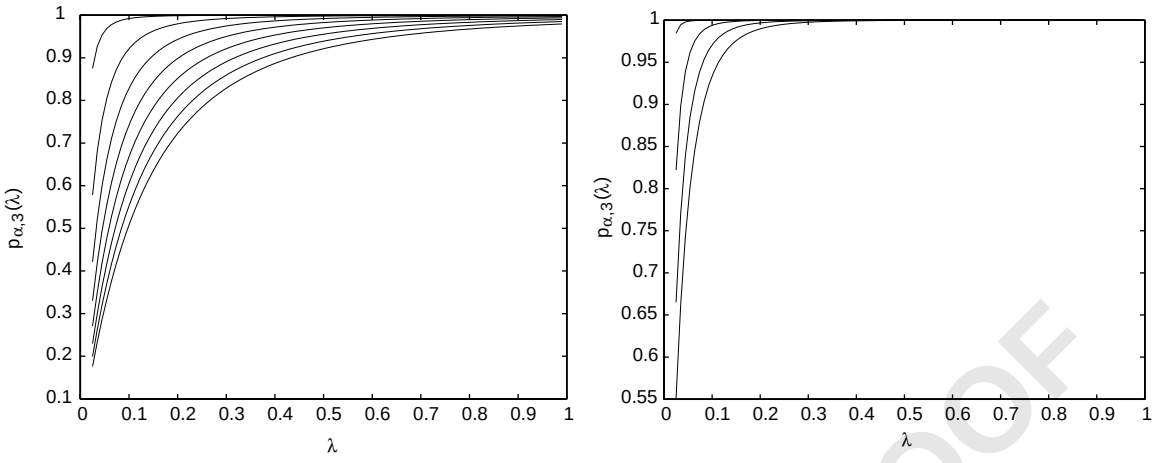

Fig. 1. The functions $\lambda s(\lambda)$ in the interval $[0.025, 1]$, for different values of α and $d = 3$ (left) and $d = 6$ (right).

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 [14] of the matrix with a small α

$$A + \alpha I = L_\alpha U_\alpha + R_\alpha, \quad M_\alpha \equiv L_\alpha U_\alpha. \tag{6}$$

Then a rational preconditioning operation used for preconditioning the original matrix A can be defined by

$$M^{-1}v = \sum_{i=1}^{d} \alpha^{i-1} M_\alpha^{-i} v. \tag{7}$$

If the matrix M_α is symmetric, then the preconditioner M is also symmetric, and can be used by the conjugate gradient algorithm. An algorithm for computing $w = M^{-1}v$ is the following.

Algorithm 2.1. Symmetric rational preconditioning operation

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

In order to relate this new technique with already known methods, consider the special case when an exact factorization is employed (i.e., $R_\alpha = 0$ in (6)) in the above algorithm. In this situation observe that $\alpha M_\alpha^{-1} = I - A M_\alpha^{-1}$ and as a result the preconditioning operation in line 3 of Algorithm 2.1 can also be written as $w := v + (I - A M_\alpha^{-1})w$. This yields an alternative algorithm which can be viewed as a generalization of Algorithm 2.1.

Algorithm 2.2. Nonsymmetric rational preconditioning operation

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

An approximate LU factorization M_α of $A + \alpha I$ can also be used in the same manner as before. However, the resulting preconditioner will no longer be symmetric in general, even if A is symmetric, but symmetry may be recovered in the usual way when M_α is available in the factored form $M_\alpha = L_\alpha L_\alpha^{\mathrm{T}}$, see, e.g., [14].

It is interesting to compare these two approaches, at least theoretically. Recall that the main idea of the rational approximation preconditioner is to approximate A^{-1} from expansions of an LU factorization of a near-by matrix, specifically $A + \alpha I$. We first consider the case when M_α represents an exact factorization of $A + \alpha I$. In that case, Algorithm 2.1 can be derived from the equality

$$A^{-1} = [(I - \alpha M_\alpha^{-1})M_\alpha]^{-1} = M_\alpha^{-1}[I - \alpha M_\alpha^{-1}]^{-1}$$

by applying a Neumann series expansion to approximately invert the bracketed term. In contrast, Algorithm 2.2 is based on approximating again A^{-1}, but through the identity

$$A^{-1} = [(I - (I - AM_\alpha^{-1}))M_\alpha]^{-1} = M_\alpha^{-1}[I - (I - AM_\alpha^{-1})]^{-1}$$

to which, again, a standard Neumann series expansion is applied to approximate the inverse of the second term of the right-hand side.

In the general case of an incomplete LU factorization, we have from (6)

$$I - AM_\alpha^{-1} = (\alpha I - R_\alpha)M_\alpha^{-1}.$$

The preconditioning matrix of Algorithm 2.1 can be written as

$$M_\alpha^{-1} \sum_{i=0}^{d-1} \alpha^i M_\alpha^{-i} = M_\alpha^{-1}(I - \alpha M_\alpha^{-1})^{-1}(I - \alpha^d M_\alpha^{-d}) \tag{8}$$

$$= (A - R_\alpha)^{-1}(I - \alpha^d M_\alpha^{-d}). \tag{9}$$

The preconditioning matrix of Algorithm 2.2 corresponds to

$$M_\alpha^{-1} \sum_{i=0}^{d-1} (I - AM_\alpha^{-1})^i = M_\alpha^{-1}(AM_\alpha^{-1})^{-1}(I - [I - AM_\alpha^{-1}]^d) \tag{10}$$

$$= A^{-1}(I - [(\alpha I - R_\alpha)M_\alpha^{-1}]^d). \tag{11}$$

As is expected, Algorithms 2.1 and 2.2 coincide only when the factorization is exact ($R_\alpha = 0$). Denote by P_1 the preconditioning matrix in (8) and by P_2 the preconditioning matrix defined by (10). The preconditioned matrices AP_i, $i = 1, 2$ are more relevant to understanding the quality of the

preconditioner. They are given by

$$AP_1 = (I - R_\alpha A^{-1})^{-1}(I - \alpha^d M_\alpha^{-d}), \tag{12}$$

$$AP_2 = I - [(\alpha I - R_\alpha)M_\alpha^{-1}]^d. \tag{13}$$

Since $\alpha I - R_\alpha = M_\alpha - A$, the preconditioned matrix AP_2 related to the second algorithm is fairly easy to analyze. Its eigenvalues are equal to $1-(1-\lambda)^d$, where λ represents a generic eigenvalue of the matrix AM_α^{-1}. In contrast, the eigenvalues of the preconditioned matrix AP_1 are not related in an easy way to those of $M_\alpha^{-1}A$, except in special situations, such as when $M_\alpha = A + \alpha I$.

We now compare formulas (12) and (13), in the general case when $R_\alpha \neq 0$. Consider first the situation when R_α is small, say much smaller than α. Then the residual matrix for the Neumann series preconditioner would be of the order of α^d in both cases, and so the accuracy of both algorithms is likely to be comparable. However, Algorithm 2.2 is more expensive than Algorithm 2.1 because of the extra matrix–vector product with A. An interesting case is when A is a dense matrix arising from electromagnetics and M_α is a sparse preconditioner to $A+\alpha I$. In this case, vector products with A are very expensive and Algorithm 2.1 is certainly advantageous.

Consider now the reverse situation when α is very small compared with R_α as measured by a certain norm. In this case, the error for Algorithm 2.1 is of the order of $\|R_\alpha\|$, while we obtain an order of $\|R_\alpha\|^d$ for Algorithm 2.2. Roughly speaking, this tells us that *when R_α is large relative to α, the use of Algorithm* 2.1 *will make little sense, because the potential gain from the expansion in α is annihilated by the large error in R_α.*

In summary, Algorithm 2.1 is likely to be competitive with the more traditional Algorithm 2.2 only when α is large enough and R_α is small. In all cases when Algorithm 2.1 is employed, the ILU factorization of $A + \alpha I$ should be fairly accurate. If not, the resulting procedure can be ineffective, a fact that is confirmed by experiments. However, it is often not a problem to compute an accurate ILU factorization of $A + \alpha I$ provided an adequate shift α is applied. All these facts are illustrated on a simple example in the next sections.

2.3. A simple illustration

In order to illustrate the points of the above discussion we now consider a test example using MATLAB. The linear system considered simulates a problem which arises in the stream-function/vorticity formulation of the Navier–Stokes equations in two dimensions. As is well known [2,7] this formulation leads to a nonlinear equation of the form

$$F(\psi) = \Delta^2\psi + Re[\psi_y(\Delta\psi)_x - \psi_x(\Delta\psi)_y].$$

Here Δ^2 represents the biharmonic operator, and u_x, u_y represents the partial derivatives of the function u with respect to x and y, respectively. The differential of the above system is represented by the linear operator:

$$\frac{\mathrm{D}F}{\mathrm{D}\psi}u \equiv \Delta^2 u + Re[(\Delta\psi)_x u_y - (\Delta\psi)_y u_x + \psi_y(\Delta u)_x - \psi_x(\Delta u)_y].$$

For the sake of simplicity let us assume that ψ_y and ψ_x are constant,[3] so the differential simplifies to

$$\frac{DF}{D\psi}u = \Delta^2 u + Re[\psi_y(\Delta u)_x - \psi_x(\Delta u)_y]. \tag{14}$$

We take a regular grid of size $nx \times ny$ with $nx = ny = 35$. The Reynolds number is set at $Re = 500$. All partial differential operators are discretized by centered differences so the biharmonic and Laplacean operators are approximated with the following 13-point and 5-point stencils:

$$\Delta^2 \approx \frac{1}{h^4}\begin{bmatrix} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{bmatrix}, \quad \Delta \approx \frac{1}{h^2}\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}.$$

The first-order differentiation corresponding to the operator $\psi_y(.)_x - \psi_x(.)_y$ which is applied to $\Delta\psi$ in (14) is also accomplished with centered differences via the stencil

$$\frac{1}{2h}\begin{bmatrix} & -\psi_x & \\ -\psi_y & 0 & \psi_y \\ & \psi_x & \end{bmatrix}.$$

The resulting Jacobian is then given by the matrix

$$A = \frac{1}{h^4}B + \frac{Re}{2h^3}E \times L,$$

where B and L are the matrices corresponding to the biharmonic and Laplacean stencils, and E corresponds to the first-order derivation stencil shown above. Normally, boundary conditions would be applied and the matrices would change accordingly. However, this is not done here. We have used the following parameters:

$$nx = ny = 35, \quad Re = 500, \quad \psi_x = -0.15, \quad \psi_y = -0.05.$$

The final matrix is scaled by h^4 and the right-hand side is set to be the vector having a value of one on the first side of the domain, corresponding to the first $nx = 35$ unknowns, and zero elsewhere.

Results for 4 different methods are shown in Fig. 2. First, a standard ILU factorization is computed using the MATLAB command `iluinc(A,droptol)` with a drop tolerance of droptol = 0.1. The resulting preconditioner is fairly unstable in this case, as is indicated from the MATLAB function command `condest(L*U)`, which gave a large value of est $= 1.88e + 11$. As one might expect GMRES(30) stagnates in this case (top curve). Then the matrix is shifted using the shift $\alpha = 1.5$ and GMRES(30) is retried with the new preconditioner. The condition number of the preconditioner now comes down to `condest(LU)=1.398e+05`. This is used as a preconditioner for GMRES(30) which results in a slight improvement but no convergence in the maximum number of steps allowed (solid curve). The same

[3] This can also be viewed as a secant method for solving the nonlinear equations, where the Jacobian is approximated by the one obtained from values of ψ_x, ψ_y at the previous Newton step.

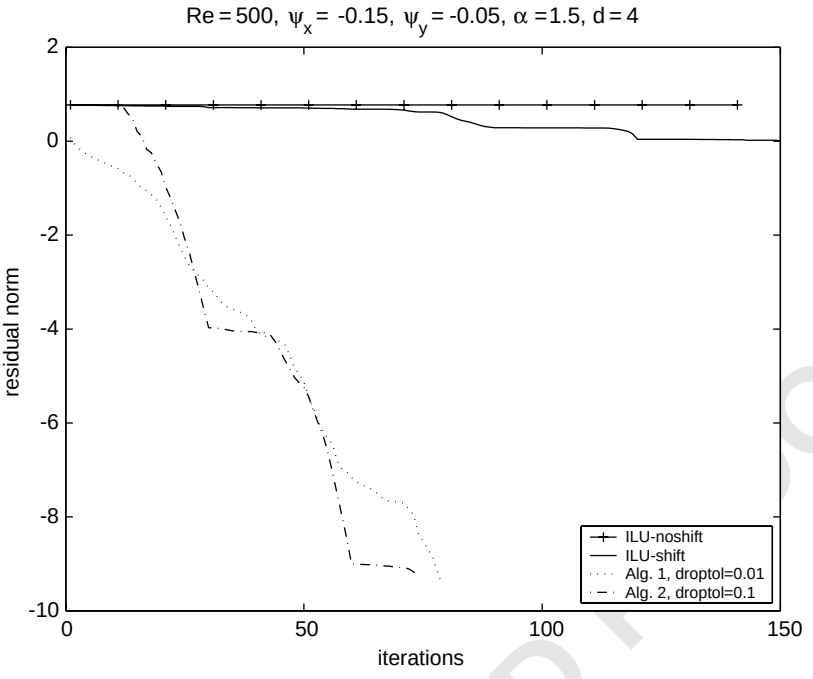

Fig. 2. Solution of the model stream function linear system of size 1225, solved by GMRES(30) preconditioned in 4 different ways.

preconditioner is now used in conjunction with Algorithm 2.1. Since the accuracy is rather poor, this method stagnates and the corresponding curve, which is not shown, is almost identical with the first (top) curve. Algorithm 2.2 on the other hand yields good convergence (dash–dotted curve). To verify the explanations given above, we performed another experiment with Algorithm 2.1 with a fairly accurate ILU factorization, one that is obtained from MATLAB with a drop tolerance droptol $= 0.01$. With this factorization, Algorithm 2.1 converges similarly to Algorithm 2.2 (dotted line). One might argue here that applying Algorithm 2.1 is more expensive. As was explained before, this is certainly not true when the matrix A is fairly dense. For this particular example, Algorithm 2.1 is in fact more advantageous than Algorithm 2.2. Indeed for droptol $= 0.1$ the total number of nonzeros for L and U together is $\text{nnz}(M) = 4761 + 3605 = 8636$ and this increases to $\text{nnz}(M) = 9303 + 8194 = 17,497$ for the more stringent droptol $= 0.01$. The number of nonzero elements in the original matrix A is $\text{nnz}(A) = 15,229$. So each of the $d - 1$ sub-steps (represented by line 3 in both algorithms) costs $17,497 + 1225 = 18,722$ operations for Algorithm 2.1 and $15,229 + 8636 + 1225 = 24,840$ operations for Algorithm 2.2. For Algorithm 2.1, when $d = 4$, this is done 3 times and another solve with M is applied in line 5 leading to a total of $18,722 \times 3 + 17,497 = 73,663$ operations. For Algorithm 2.2, this count becomes $24,840 \times 3 + 8636 = 83,156$. Clearly, the advantage for Algorithm 2.1 improves as d increases. Another point to make here is that this problem is actually not a hard one to solve. A good ILU factorization is obtained for the nonshifted matrix when a small enough drop tolerance is used.

2.4. Inner–outer rational preconditioning

An appealing alternative to the above approach aims at extracting an optimal solution from the iterates of Algorithm 2.1. Instead of a preconditioning of the form (7) we may 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 $\|v - Aw\|_2$. In the case when M does not represent an accurate ILU factorization, we already know that such a sequence is not likely to be effective. It is therefore more general and effective to seek an optimal combination of iterates from Algorithm 2.2. This means that we seek a combination of the vectors of the preconditioned Krylov subspace

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

The usual least-squares solution obtained by GMRES is calculated to minimize the residual norm. This method is nothing but an inner–outer GMRES iteration, in which the inner solve is itself a GMRES iteration using a Krylov subspace of dimension d. In this case, the preconditioner is modified at each step, hence a flexible accelerator such as FGMRES [12] must be used. Numerical results indicate that this is more costly but often more effective than the simple expansion (7) (see Fig. 5).

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 (9) with $R_\alpha = 0$, the preconditioning matrix which is defined by Algorithm 2.1 is equal to

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

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 (4) (hence Fig. 1 plots also the function $1 - \rho_i$). 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 any eigenvalue outside the disk $\bar{D}(-\alpha, \alpha)$ centered at $-\alpha$ and of radius α, will be damped, i.e., it will be transformed into

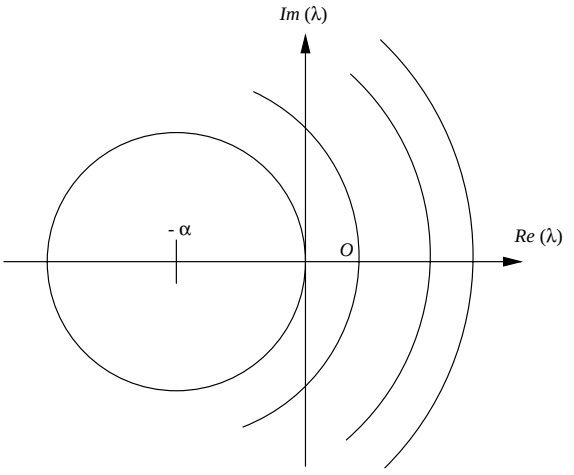

Fig. 3. Damping region for preconditioner is outside the disk.

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 λ_i is 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 Fig. 3 show the lines where the eigenvalues have the same damping factor ρ. If α is not changed during the iteration process, then the eigencomponents 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) are likely to change very little. 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. Tests with inner–outer techniques and variable shift

We now go back to the example of Section 2.3 to illustrate the techniques of the previous two sections. For the variable shift strategy of Section 2.5, we do not use a different factorization for each different α, since this would be prohibitive. Instead, the same LU factorization as for the other methods is used, i.e., the one obtained with the shift $\alpha = 1.5$ and using the same drop tolerance of 0.01. The variable shift strategy is used in conjunction with Algorithm 2.2. The variation of α occurs in line 3 which is replaced by $w := v + (I - (A + \sigma I)M_\alpha^{-1})w$. The resulting preconditioner becomes

$$(A + \sigma I)^{-1}(I - [([\alpha - \sigma]I - R_\alpha)M_\alpha^{-1}]^d)$$

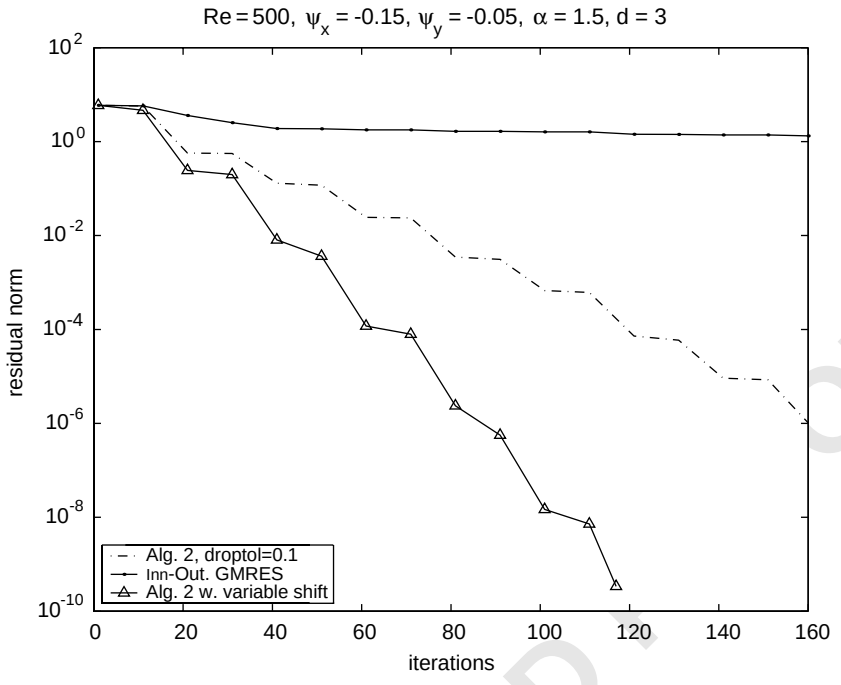

Fig. 4. Solution of the model stream function linear system with $Re = 500$.

instead of (see Eq. (11))

$$A^{-1}(I - [(\alpha I - R_\alpha)M_\alpha^{-1}]^d).$$

Hence α has been changed to $\alpha - \sigma$ in the right parentheses, with $0 < \sigma < \alpha$, thus (partially) decreasing α as proposed in Section 2.5. Regarding the shifting strategy, the initial shift σ is set to $\sigma = 0.05$ then at each FGMRES outer iteration σ is increased by $\delta\sigma \equiv 0.01$. Other strategies for shifting have been tested but did not yield better results.

Fig. 4 compares three different methods: Algorithm 2.2, inner–outer GMRES of Section 2.4 and the variable-α procedure of Section 2.5. In order to have shorter restarts, the Krylov subspace dimension was reduced from $m = 30$ to 20. All other parameters remain the same as in the example of Section 2.3. In particular, the shift α is again 1.5, and the drop tolerance for `iluinc` is droptol$=0.1$, yielding the same ILU factorization. The degree d has been set to $d = 3$. This is also the subspace dimension for the inner GMRES iteration in inner–outer GMRES algorithm. In this particular case, the inner–outer GMRES iteration does not converge. Algorithm 2.2 (with fixed shift) behaves similarly to the case with $m = 30$ shown in Section 2.3. The interesting observation is that the variable shift Algorithm does much better than the other 2 algorithms. Note that this improvement is achieved with a small change to Algorithm 2.2, since in most iterations the shift σ is rather small ! Observe also that this method uses about the same number of operations per iteration as Algorithm 2.2 (without shift) and fewer than the inner–outer GMRES method. Although only iteration counts are shown, the plots in Figs. 4 and 5 do provide some idea of the operations cost in some cases.

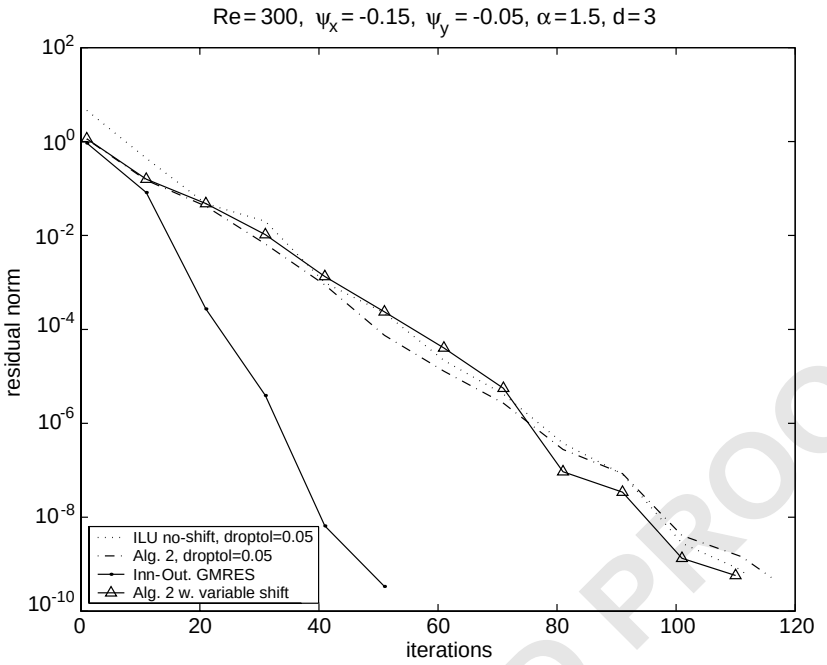

Fig. 5. Solution of the model stream function linear system with $Re = 300$.

As was mentioned above, the inner–outer GMRES iteration did not perform too well in this example. In our experience we found that this happened occasionally but that this occurrence is by no means general. For example, in the next test we changed the Reynolds number to $Re = 300$ and computed the ILU factorization with a better accuracy using droptol $= 0.05$. In that case, we had `condest(LU)=3.1e+05` without shift and `condest(LU)=138` with shift. The ILU factorization for the matrix A (without shift) is accurate and fairly stable—so the corresponding ILU preconditioning works well in this case. The corresponding run is shown in a dotted line in Fig. 5. As is expected in this case not much can be gained from the rational approximation preconditioner. In particular, the variable shift technique is no longer competitive with the inner–outer GMRES technique which does quite well.

3. Analysis in the case of an exact factorization

This section gives some error bounds for the CG algorithm applied to the solution of the preconditioned system

$$Bx := M^{-1}Ax = M^{-1}b. \tag{15}$$

The rational transformation $B = r(A)$ makes the spectrum of B clustered around 1, and the usual error bound for the CG algorithm can be improved, both for the first iterations and asymptotically.

Here it is assumed that the LU factorization is exact, i.e., $R_\alpha = 0$ in (6). As was already seen, this means that Algorithms 2.1 and 2.2 coincide. The case of an approximate LU factorization is

more difficult to analyze, though one can expect the behavior to be similar if the perturbation of the exact factorization remains small. As was mentioned in Section 2.2 and observed in the numerical experiments in Sections 2.3 and 2.6, the ILU factorization of $A + \alpha I$ should be fairly accurate in order to obtain an effective procedure.

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

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

associated to normalized eigenvectors v_i, $i = 1, \ldots, n$. The preconditioning operation corresponding to (3) is modified by a constant β into

$$r(\lambda) = \frac{1}{\beta}\left[1 - \left(\frac{\alpha}{\lambda + \alpha}\right)^d\right], \quad \beta = 1 - \left(\frac{\alpha}{1 + \alpha}\right)^d, \tag{16}$$

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

$$\mu_i = r(\lambda_i), \quad 1 \leqslant i \leqslant n. \tag{17}$$

The choice of β ensures that $\mu_n = r(\lambda_n) = 1$. As the matrix A is assumed 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 mth step of the CG algorithm applied to the matrix A, and by y_m the approximate solution obtained with the matrix B, starting with $y_0 = x_0$. A well-known optimality property of the CG algorithm, states that

$$\|y_m - x\|_B^2 = \min_{p \in \mathbb{P}_m,\ p(0)=1} \|p(B)(x_0 - x)\|_B^2, \tag{18}$$

where $\mathbb{P}_m$ is the set of polynomials of degree at most m, and $\|.\|_B$ is the B-norm defined by $\|z\|_B^2 = (Bz, z)$. The minimizer p_m of (18) 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

$$\|x_m - x\|_A^2 \leqslant 4\left(\frac{1 - \sqrt{\lambda_1}}{1 + \sqrt{\lambda_1}}\right)^{2m} \|x_0 - x\|_A^2, \tag{19}$$

whereas the error for the preconditioned system (15) is

$$\|y_m - x\|_B^2 \leqslant 4\left(\frac{1 - \sqrt{\mu_1}}{1 + \sqrt{\mu_1}}\right)^{2m} \|x_0 - x\|_B^2. \tag{20}$$

3.1. Error bounds for the first iterations

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

$$p(t) = \frac{T_m[1 + 2(\mu_{k+1} - t)/(1 - \mu_{k+1})]}{T_m[1 + 2\mu_{k+1}/(1 - \mu_{k+1})]} \tag{21}$$

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., [14]:

$$p^2(t) \leqslant 1, \quad \forall t \in [0, 1], \tag{22}$$

$$p^2(t) \leqslant \zeta := 4\left(\frac{1-\sqrt{\mu_{k+1}}}{1+\sqrt{\mu_{k+1}}}\right)^{2m} \quad \forall t \in [\mu_{k+1}, 1]. \tag{23}$$

We have

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

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 two parts as follows:

$$\|x_0 - x\|_B^2 = S_1 + S_2,$$

$$S_1 = \sum_{i=1}^{k} e_i^2 \mu_i, \quad S_2 = \sum_{i=k+1}^{n} e_i^2 \mu_i.$$

Then, using (22) and (23), we obtain

$$\|p(B)(x_0 - x)\|_B^2 = \sum_{i=1}^{k} p^2(\mu_i) e_i^2 \mu_i + \sum_{i=k+1}^{n} p^2(\mu_i) e_i^2 \mu_i \leqslant S_1 + \zeta S_2,$$

from which it follows that

$$\|y_m - x\|_B^2 \leqslant S_1 + 4\left(\frac{1-\sqrt{\mu_{k+1}}}{1+\sqrt{\mu_{k+1}}}\right)^{2m} \|x_0 - x\|_B^2. \tag{24}$$

Observe that the special choice $k = 0$ leads to $S_1 = 0$ which yields inequality (20) as a particular case.

Up to now, we have not exploited the clustering of the spectrum of $B = r(A)$ around 1. A consequence of this clustering is that $S_1 = S_1(k)$ may be small while at the same time μ_{k+1} is 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 must keep in mind that we want to use an α larger than λ_1 but smaller than 1, which gives the possible range for c. We now define L to be the smallest scalar independent of α and k, for which

$$\sum_{i=1}^{k} e_i^2 \leqslant \lambda_k L \|x_0 - x\|^2 \quad \forall k = 1, 2, \ldots, n. \tag{25}$$

Recalling that $\|x_0 - x\|^2 = \sum_{i=1}^{n} e_i^2$, note that the linear function $l(\lambda) \equiv L\lambda$ can be viewed as an upper bound for the density function, a step function whose value at each eigenvalue λ_j is defined by

$$\phi(\lambda_j) = \frac{\sum_{i=1}^{j} e_i^2}{\sum_{i=1}^{n} e_i^2}.$$

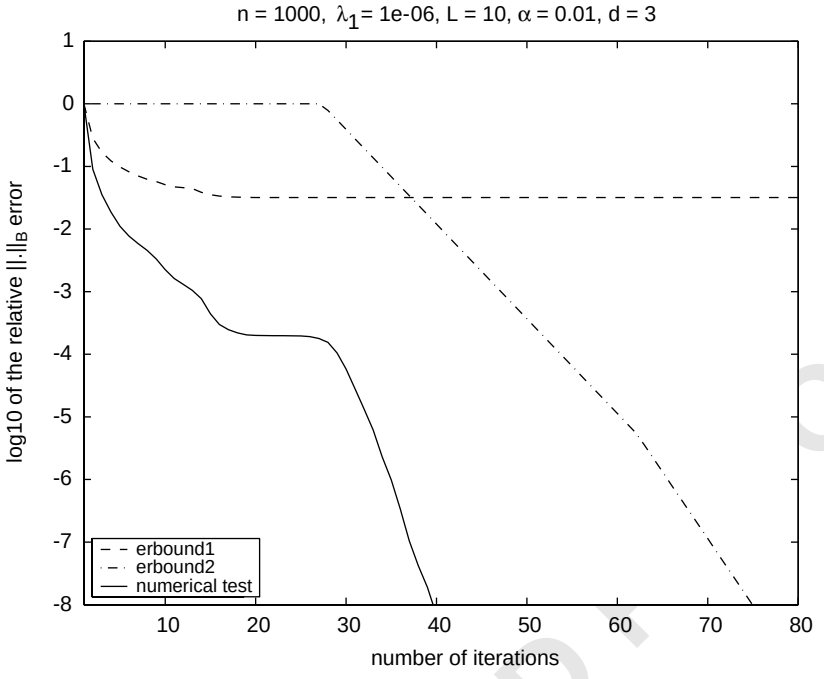

Fig. 6. Error bounds $\|y_m - x\|_B$.

For example, if the matrix A has a uniform spectrum $\lambda_i = i/n$ and if $e_i^2 = 1$ for all i, then (25) holds for $L = 1$. A large value of L corresponds to a clustering of the spectrum of A near 0, or to an initial error $x_0 - x$ essentially concentrated on the eigenspace associated with 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 (see Figs. 6 and 7).

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*

$$\|y_m - x\|_B^2 \leqslant \left[c\alpha L + 4 \left(\frac{1}{\sqrt{(c+1)^d} + \sqrt{(c+1)^d - 1}} \right)^{4m} \right] \|x_0 - x\|^2. \tag{26}$$

Proof. Since $\mu_i \leqslant 1$ and by assumption (25), we have

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

It follows from (24), $\lambda_k \leqslant \lambda_{k+1} = c\alpha$, $\|\cdot\|_B \leqslant \|\cdot\|$ and $\mu_{k+1} = r(\lambda_{k+1})$ that

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

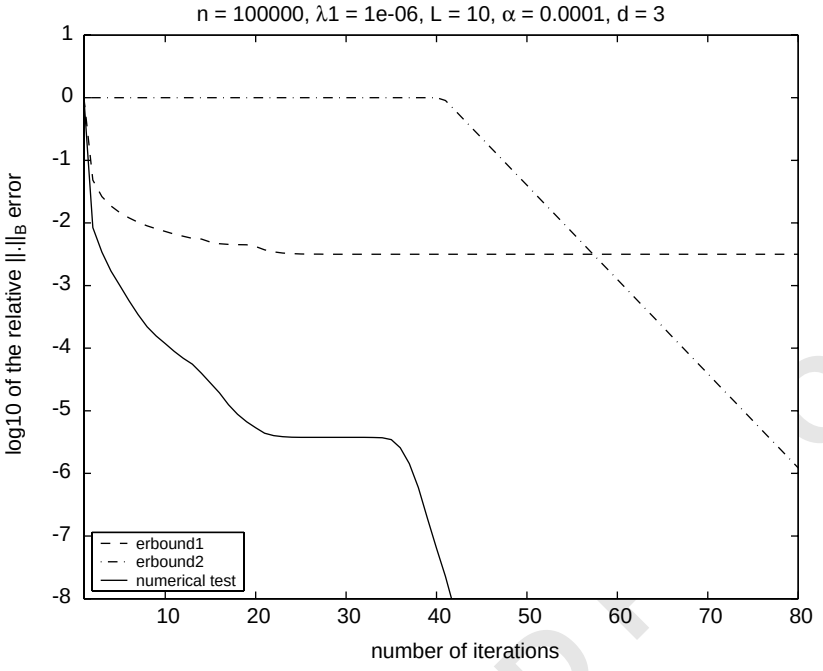

Fig. 7. Error bounds $\|y_m - x\|_B$.

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

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

then, using

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

we obtain

$$\frac{1-\sqrt{r(c\alpha)}}{1+\sqrt{r(c\alpha)}} \leqslant \left(\frac{\sqrt{1/(c+1)^d}}{1+\sqrt{1-1/(c+1)^d}}\right)^2 = \left(\frac{1}{\sqrt{(c+1)^d}+\sqrt{(c+1)^d-1}}\right)^2. \tag{27}$$

which completes the proof. □

We can notice that if we use the polynomial

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

instead of (21), then the following bound is obtained for $1 \leqslant j \leqslant m$:

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

However, for large m, the following section shows 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:

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

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

$$\max_{\mu_j \in \Lambda(B)} |C_m(\mu_j)| \leqslant \max_{j=k+1,\ldots,n} \prod_{i=1}^{k}\left|\frac{\mu_i - \mu_j}{\mu_i}\right| \times \frac{1}{T_{m-k}[1 + 2(\mu_{k+1})/(\mu_n - \mu_{k+1})]}. \tag{28}$$

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 $m \geqslant k$, we have*

$$\|y_m - x\|_B \leqslant \frac{2\kappa^k(\alpha)}{[\sqrt{(c+1)^d} + \sqrt{(c+1)^d - 1}]^{2(m-k)}}\|x_0 - x\|_B, \tag{29}$$

where

$$\kappa(\alpha) = \frac{1 - ((\lambda_1 + \alpha)/(1+\alpha))^d}{((\lambda_1 + \alpha)/\alpha)^d - 1}. \tag{30}$$

Proof. Consider first the product term

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

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

$$\kappa(\alpha) = \frac{1 - \frac{1}{\beta}(1 - \alpha^d/(\lambda_1 + \alpha)^d)}{\frac{1}{\beta}(1 - \alpha^d/(\lambda_1 + \alpha)^d)}$$

$$= \frac{1 - \alpha^d/(1+\alpha)^d - 1 + \alpha^d/(\lambda_1+\alpha)^d}{1 - \alpha^d/(\lambda_1+\alpha)^d}$$

$$= \frac{\alpha^d/(\lambda_1+\alpha)^d - \alpha^d/(1+\alpha)^d}{1 - \alpha^d/(\lambda_1+\alpha)^d}.$$

Multiplying numerator and denominator by $(\lambda_1+\alpha)^d/\alpha^d$ yields (30). Next, the second term is bounded by

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

and, using (27) once more, we obtain for $\mu_{k+1} = r(c\alpha)$

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

which completes the proof. □

A fair comparison would be between one step of the rational preconditioner versus 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,

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

The above asymptotic argument ignores the potentially large constant in the numerator of (29). However, it gives a rough comparison of the situation at the asymptotic regime where, for example, a large accuracy is required.

The estimated error bounds resulting from Propositions 3.1 and 3.2 are illustrated by the curves `erbound1` and `erbound2` in Figs. 6 and 7 for two sets of parameters: $d = 3$, $\lambda_1 = 10^{-6}$, $L = 10$, $n = 10^3$, $\alpha = 10^{-2}$ in Fig. 6, and $d = 3$, $\lambda_1 = 10^{-10}$, $L = 10$, $n = 10^5$, $\alpha = 10^{-4}$ in Fig. 7. The other eigenvalues were chosen as follows:

$$\lambda_k = \max(\lambda_1 + (k-1)/(nL), (2k-n)/n), \quad 2 \leqslant k \leqslant n.$$

The two curves `erbound1` and `erbound2` complement each other since the first estimate is better than the second one for the earlier iterates. They are compared to numerical results obtained from the CG algorithm preconditioned with Algorithm 2.1. The matrix A is of the form $A = Q^{\mathrm{T}}\Lambda Q$ where Q is a randomly chosen sparse orthogonal matrix and Λ is diagonal with $\Lambda_{kk} = \lambda_k$. The factorization was not completely performed: we used the MATLAB command `U = cholinc(A+alpha*I,droptol)` with `droptol` $= \alpha^2$.

Table 1
Description of test problems

Name	n	n_z	Dominance (%)	Symmetry	Matrix source
ELTCOQUE	38,002	949,452	0.6	Yes	Shell modeling
MCHLNF	49,800	4,136,484	5	Yes	Tire design
MCHLNE	49,800	4,136,580	4.6	No	Tire design

4. Rational acceleration for realistic problems

This section reports on a few numerical experiments (on a DEC Alpha) 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 shifting and large fill-in may be needed to achieve convergence as reported in [16] for tire design problems. However, choosing the best shift value can be time consuming, and the preconditioning becomes quite expensive to apply in the case of large fill-in. We attempt to show how rational preconditioning (Algorithm 2.1 or 2.2) can handle these difficulties. Since the purpose of the experiments shown here is to solve realistic applications problems more efficiently using a form of rational approximation, we present the experimental results for either Algorithm 2.1 or 2.2 depending on which algorithm we found to perform better for a particular problem. For a comparison of the two algorithms, see Section 2.3.

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 Americas Research and Development Corporation. 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`.

Restarted GMRES was used as the accelerator. Specifically, the FGMRES(k) variant, which allows variable preconditioning [14], was employed in cases when the preconditioner changes during iteration. Deflated GMRES(k) [3,9] was used in 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. For these problems, the effects of the artificial right-hand side

and the one coming from the application have been studied in [16]. It has been observed that the artificial right-hand side does not make these problems easier and that both right-hand side types result in a similar convergence pattern.

Rational acceleration is applied to the factorization produced by the algebraic recursive multilevel solver (ARMS) [15]. 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 (say, 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 are properly permuted by setting the independent set tolerance in ARMS to 1, see [15].

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 sufficient 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

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

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).

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. In our experiments, we have observed that, for this problem, halving the amount of fill further makes the preconditioner very inaccurate, and the solution time grows along with the added cost of increased degree d. Hence, a reasonable increase of the degree and reduction in 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

Here, 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 $\mathscr{E} = \log(\|(LU)^{-1}\|_{\inf})$ was shown and was suggested as a practical means of evaluating the quality of a preconditioner. We can inexpensively compute $\mathscr{E}_\alpha$ as

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

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. Fig. 8 shows the convergence curves for different choices of α with the lowest degree 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 ($\mathscr{E}_\alpha = 0.29$). It is possible to start with some large α (say, 0.8) and gradually decrease it as long as the indicator $\mathscr{E}_\alpha$ stays small. Changing α dynamically requires modifying the

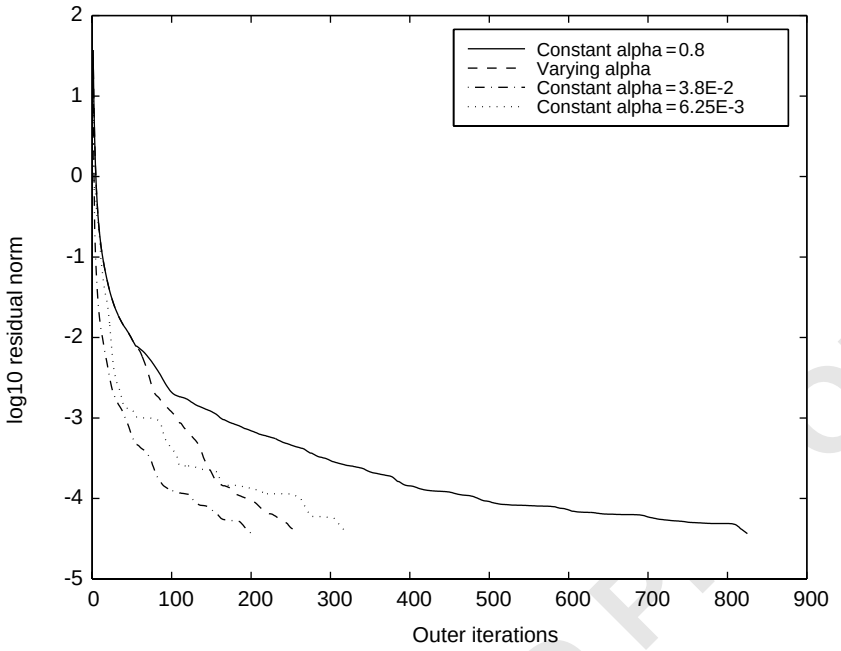

Fig. 8. Choosing α for the problem ELTCOQUE without rational approximation.

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 re-factor 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 Fig. 8 indicates that the iterative convergence is much faster for varying α dynamically than for some constant *large* α (solid line). In this case, it is faster not only in terms of iteration numbers as shown in Fig. 8 but also in terms of the execution times, which include refactoring for varying α. The solution times are 222.22 and 303.54 s, respectively.

Monitoring $\mathscr{E}_\alpha$ allows an early detection of a possible preconditioner instability for some small α indicating that it should not be decreased further. We can also use $\mathscr{E}_\alpha$ to find a better constant shift and re-construct a preconditioner with this shift. The dash–dotted and dotted curves show the convergence histories for the two constant shift values ($3.8 \cdot 10^{-2}$ and $6.25 \cdot 10^{-3}$, respectively), which are obtained at the points immediately preceding a sharp increase in the estimate of $\mathscr{E}_\alpha$ for two different strategies of decreasing α, fast and gradual, respectively. In particular, a fast decrease, which consists of halving α at each restart, quickly reaches a shift value corresponding to an unstable preconditioner. However, the preconditioner re-constructed with the constant shift value $6.25 \cdot 10^{-3}$ obtained from this fast strategy leads to a slower convergence than the one with the shift $3.8 \cdot 10^{-2}$ obtained from a more gradual α decrease. In fact, the constant shift $3.8 \cdot 10^{-2}$ seems to give the best convergence among the four shift choices. However, finding this optimal shift value is expensive.

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

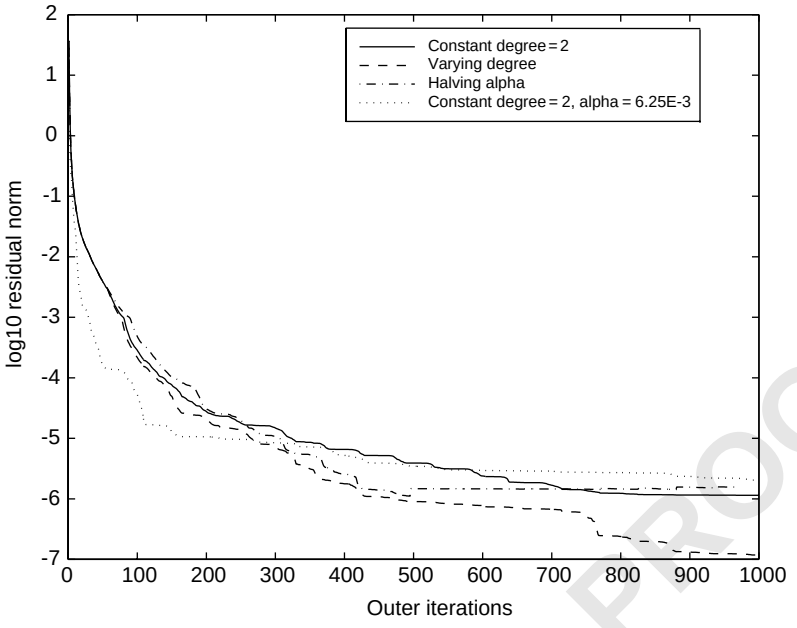

Fig. 9. Achieving high accuracy for the problem ELTCOQUE with rational approximation.

to achieve maximum accuracy in 1000 iterations given four ways to choose (α, d) in the rational approximation (Algorithm 2.2) 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 $(6.25 \cdot 10^{-3})$ as given in Fig. 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.

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 can sometimes be 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 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 helpful

in improving 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, Vol. 3: coques, Hermès, Paris, 1992.
[2] P.N. Brown, Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, SIAM J. Sci. Statist. Comput. 11 (1990) 450–481.
[3] A. Chapman, Y. Saad, Deflated and augmented Krylov subspace techniques, Numer. Linear Algebra Appl. 4 (1997) 43–66.
[4] E. Chow, Y. Saad, Experimental study of ILU preconditioners for indefinite matrices, J. Comput. Appl. Math. 86 (1997) 387–414.
[5] E. Chow, Y. Saad, Approximate inverse preconditioners via sparse-sparse iterations, SIAM J. Sci. Comput. 19 (1998) 995–1023.
[6] H.C. Elman, A stability analysis of incomplete LU factorizations, Math. Comput. 47 (1986) 191–217.
[7] R. Glowinski, H.B. Keller, L. Reinhart, Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems, SIAM J. Sci. Statist. Comput. 6 (1985) 793–832.
[8] T.A. Manteuffel, An incomplete factorization technique for positive definite linear systems, Math. Comput. 34 (1980) 473–497.
[9] R.B. Morgan, A restarted GMRES method augmented with eigenvectors, SIAM J. Matrix Anal. Appl. 16 (1995) 1154–1171.
[10] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[11] A. Ruhe, Rational Krylov sequence methods for eigenvalue computations, Linear Algebra Appl. 58 (1984) 391–405.
[12] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Statist. Comput. 14 (1993) 461–469.
[13] Y. Saad, Theoretical error bounds and general analysis of a few Lanczos-type algorithms, in: J.D. Brown, M.T. Chu, D.C. Ellison, R.J. Plemmons (Eds.), Proceedings of the Cornelius Lanczos International Centenary Conference, SIAM, Philadelphia, PA, 1994, pp. 123–134.
[14] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, New York, 1996.
[15] Y. Saad, 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.
[16] M. Sosonkina, J.T. Melson, Y. Saad, L.T. Watson, Preconditioning strategies for linear systems in tire design, Numer. Linear Algebra Appl. 7 (2000) 743–757.