ELSEVIER Journal of Computational and Applied Mathematics 59 (1995) 91–107

JOURNAL OF
COMPUTATIONAL AND
APPLIED MATHEMATICS

A polynomial preconditioner for the GMRES algorithm

M.B. van Gijzen[1]

Mathematisch Instituut, Utrecht University, P.O. Box 80 010, 3508TA Utrecht, Netherlands

Received 6 July 1993; revised 17 November 1993

Abstract

The major drawback of GMRES is that the storage demands and the number of operations per iteration increase with the number of iterations. It is important to avoid that so many iterations are needed that the work per iteration and the storage requirements become unacceptably high. This paper describes a polynomial preconditioner with which this can be achieved efficiently. The polynomial preconditioner is constructed so that it has a minimization property in an area of the complex plane. A suitable area, and hence the preconditioning polynomial, can be obtained from eigenvalue estimates. The polynomial preconditioner is very simple and easy to implement.

Keywords: Preconditioning; GMRES; Ritz values; Nonsymmetric linear systems

1. Introduction

The well-known and popular conjugate gradient method [8] for solving symmetric positive-definite linear systems of equations is a method that is both simple and optimal. By simple we mean that the approximate solution vectors can be obtained by short recursions. Only a limited number of vectors need to be stored. The method is optimal in the sense that in every iteration the error in the solution vector is minimized in some norm over the Krylov subspace. Conjugate Gradient-like methods that are based on short recursions also exist for nonsymmetric systems. Examples are Bi-CG [5], CGS [15] and Bi-CGSTAB [17]. A serious drawback of these methods is that they lack a minimization property. A class of methods that does have a minimization property is based on the Arnoldi method [2] for reducing a general square matrix to Hessenberg form. The most popular and widely used method of this kind is the generalized minimal residual algorithm (GMRES) of Saad and Schultz [13]. In this method the residual is explicitly minimized over the Krylov subspace in every iteration. As a consequence, a basis of this subspace must be stored. Therefore the storage requirements grow linearly with the number of iterations. Since this basis is orthogonalized, the computational complexity even grows quadratically! Already in [13] it was

[1] This work has been performed at TNO Building and Construction Research, with support of the Delft University of Technology.

SSDI 0377-0427(94)00015-S

proposed to restart the algorithm after a cycle of iterations. This obviously fixes the maximum amount of storage and operations needed in an iteration. The convergence of the cyclic variant GMRES, however, cannot be proved. It is often observed that due to the restarts the super linear convergence behaviour of GMRES is lost, or even that no convergence occurred at all. The cyclic variant of GMRES is often denoted by GMRES(m), with m the number of iterations in a cycle. GMRES without restart is also denoted by full GMRES. As an alternative several authors have proposed to combine GMRES with another, more simple, iterative method [4, 10, 12, 14]. The strategy they propose is to perform only a limited number of GMRES iterations. Then the so-called Ritz values, approximations for the eigenvalues, are computed. From this information suitable iteration parameters are determined for, for example, the Chebychev iterative method or Richardson's method. These methods are then applied in the second stage of the solution phase. The combination of GMRES with another method is denoted by a hybrid GMRES algorithm. Such algorithms will be discussed in the next section. Hybrid GMRES methods are closely related to polynomial preconditioners. The Ritz values can also be used for computing parameters for a polynomial preconditioner. This polynomial preconditioner can then be applied in every iteration. The idea is to reduce the number of iterations needed, at the expense of performing more operations per iteration. As a consequence of the reduction of the number of iterations the storage requirements are lower, and, hopefully, so is the computational complexity. In Section 3, a polynomial preconditioner will be discussed that can be used for this purpose. Until recently the Ritz values were exploited for computing iteration parameters for the second stage of the hybrid method. However, in [10] it was suggested that the zeroes of the GMRES residual polynomial might be better suited for this purpose. Section 4 discusses an algorithm for computing these zeroes. Section 5 describes the construction of a contour that encloses the spectrum. Section 6 describes numerical experiments.

2. Hybrid GMRES algorithms

GMRES has originally been proposed in [13]. Their version is based upon Arnoldi's method [2], which constructs an orthonormal basis of the Krylov subspace $K^m(A; \boldsymbol{v}_1)$. This subspace is defined by

$$K^m(A; \boldsymbol{v}_1) = span\{\boldsymbol{v}_1, A\boldsymbol{v}_1, \ldots, A^{m-1}\boldsymbol{v}_1\}. \tag{2.1}$$

Arnoldi's method can be described as follows.

```
Let $\boldsymbol{v}_1$ be given with $\|\boldsymbol{v}_1\| = 1$
$m = 1, 2, 3, \ldots$
    $h_{i,m} = \boldsymbol{v}_i^T A \boldsymbol{v}_m, \quad i = 1, 2, \ldots, m$
    $\boldsymbol{w} = A\boldsymbol{v}_m - \sum_{i=1}^{m} h_{im}\boldsymbol{v}_i$          (2.2)
    $h_{m+1,m} = \|\boldsymbol{w}\|$
    if $h_{m+1,m} = 0$ then stop else $\boldsymbol{v}_{m+1} = \boldsymbol{w}/h_{m+1,m}$
end m
```

Let V_m be the $n \times m$ matrix, which has $\boldsymbol{v}_1, \dots, \boldsymbol{v}_m$ as its columns. Formally the process (2.2) can then be described by

$$A V_m = V_{m+1} \boldsymbol{H}_m, \tag{2.3}$$

where the $(m+1) \times m$ matrix $\boldsymbol{H}_m$ is given by

$$\boldsymbol{H}_m = \begin{bmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,m} \\ h_{2,1} & h_{2,2} & \dots & h_{2,m} \\ & h_{3,2} & \dots & h_{3,m} \\ & & \dots & \\ & \mathbf{0} & \dots & h_{m,m} \\ & & & h_{m+1,m} \end{bmatrix}. \tag{2.4}$$

The idea of GMRES is to find a correction vector $z_m \in K^m(A, \boldsymbol{r}_0)$ that minimizes the expression

$$\min_{z_m} \|\boldsymbol{b} - A(\boldsymbol{x}_0 + z_m)\| = \min_{z_m} \|\boldsymbol{r}_0 - A z_m\|. \tag{2.5}$$

As the first basis vector of the Krylov space we take $\boldsymbol{v}_1 = \boldsymbol{r}_0 / \|\boldsymbol{r}_0\|$. The other basis vectors are generated by the Arnoldi process. Since $z_m \in K^m(A, \boldsymbol{r}_0)$, z_m can be written as

$$z_m = V_m \boldsymbol{y}_m \tag{2.6}$$

with $\boldsymbol{y}_m$ a vector of dimension m. Using (2.3) leads to

$$\begin{aligned} \min_{z_m} \|\boldsymbol{b} - A(\boldsymbol{x}_0 + z_m)\| &= \min_{\boldsymbol{y}_m} \|\boldsymbol{r}_0 - A V_m \boldsymbol{y}_m\| \\ &= \min_{\boldsymbol{y}_m} \|\boldsymbol{r}_0 - V_{m+1} \boldsymbol{H}_m \boldsymbol{y}_m\| \\ &= \min_{\boldsymbol{y}_m} \| \|\boldsymbol{r}_0\| \boldsymbol{e}_1 - \boldsymbol{H}_m \boldsymbol{y}_m\|, \end{aligned} \tag{2.7}$$

where $\boldsymbol{e}_1$ is the first unit vector of dimension $m+1$. Because $\boldsymbol{H}_m$ is almost an upper Hessenberg matrix, the minimal $\boldsymbol{y}_m$ can be rather easily determined, e.g., by factoring $\boldsymbol{H}_m$ as $\boldsymbol{H}_m = \boldsymbol{Q}_m \boldsymbol{R}_m$, in which $\boldsymbol{Q}_m$ is a product of Givens rotations and $\boldsymbol{R}_m$ is upper triangular. Note that

$$\tilde{\boldsymbol{H}}_m = V_m^{\mathrm{T}} A V_m \tag{2.8}$$

is an upper Hessenberg matrix that consists of the first m rows of $\boldsymbol{H}_m$. The eigenvalues of this matrix, known as Ritz values, converge to the eigenvalues of A.

The number of operations and the storage requirements of the Arnoldi process grow with the iteration number. Therefore several authors have proposed to perform only a limited number of GMRES iterations. Then iteration parameters and a good starting vector for another, more simple iterative process are computed. For example, Elman, Saad and Saylor proposed [4] to take advantage of the eigenvalue estimates that can be obtained by computing the eigenvalues of $\boldsymbol{H}_m$, by constructing an ellipse that encloses the spectrum of the coefficient matrix. If such an ellipse is known the Chebychev iterative method for nonsymmetric systems, described in [9], can be applied.

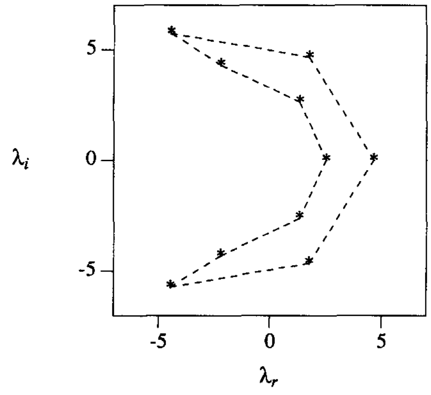

Fig. 1. Piecewise linear contour enclosing the spectrum.

Smolarski and Saylor observed [14] that sometimes the spectrum of eigenvalues is very poorly approximated by an ellipse. This is for example the case if the spectrum is "boomerang"-shaped. Therefore they proposed to take a piecewise linear polygon as a contour enclosing the spectrum. Fig. 1 gives an example of such a polygon.

The contour can be constructed in several ways. A simple strategy will be discussed in Section 4. On the contour a residual polynomial, i.e., a polynomial R that adheres to the condition $R(0) = 1$, is sought that is minimal in some norm. This approach is based on the maximum modulus lemma [1] that implies that the maximum of modulus of the polynomial R on the domain enclosed by the polygon is smaller than or equal to the maximum of the modulus of R on the polygon. From R optimum parameters are computed for a Richardson iterative method. This Richardson method is applied in the second stage of the solution scheme.

The approach based on the maximum modulus lemma has a serious drawback. If the ellipse or polygon encloses the origin then the maximum of the modulus of the residual polynomial cannot be smaller than 1, since the modulus of this polynomial is 1 in the origin. In this case hybrid methods fail to converge in the second stage of the solution scheme and therefore cannot be employed.

3. Derivation of a least squares polynomial preconditioner

The hybrid methods discussed in the previous section work along the following line:

Perform a limited number of GMRES iterations applied to $\boldsymbol{Ax} = \boldsymbol{b}$
(first stage)
↓
Compute the Ritz values → compute parameters for Richardson's or Chebychev's method
↓
Apply Richardson's or Chebychev's method to $\boldsymbol{Ax} = \boldsymbol{b}$ with initial guess from the first stage
(second stage)

The iterative methods applied in the second stage of the hybrid GMRES methods can of course also be applied as polynomial preconditioners for GMRES. We then get a method with the following structure:

$$\begin{tikzcd}
\text{Perform a limited number of GMRES iterations applied to } \boldsymbol{Ax} = \boldsymbol{b} \text{ (first stage)} \arrow[d] \\
\text{Compute the Ritz values} \rightarrow \text{compute parameters for preconditioning polynomial } P_n(A). \arrow[d] \\
\text{Apply GMRES to } \boldsymbol{A}P_n(\boldsymbol{A})\boldsymbol{y} = \boldsymbol{b},\ \boldsymbol{x} = P_n(\boldsymbol{A})\boldsymbol{y} \text{ with the initial guess from the first stage (second stage)}
\end{tikzcd}$$

The polynomials needed for the second stage of a hybrid method must be of a high degree, i.e., of degree of order 20 or more, since many iterations may have to be performed in the second stage. The degree of the preconditioning polynomial can be much lower, of degree 5 or less, of degree 10 at most, because the goal of the preconditioner is to reduce the number of GMRES iterations and a reduction of a factor 4 or 5 will often be sufficient. Numerical stability will therefore be less troublesome than for hybrid methods. Therefore, if we had the choice between stability and simplicity we would have given more weight to simplicity in the construction of the polynomial preconditioner. Although the derivation of the least squares polynomial preconditioner can be easily extended to the case of general complex matrices, we will confine ourselves to matrices with real entries only.

Let z be complex and let the polynomials of a degree i, Q_i $(i = 0, n)$ form a basis of the space of polynomials of degree less than or equal to n. Let

$$P_n(z) = \sum_{i=0}^{n} \alpha_i Q_i(z) \tag{3.1}$$

be a polynomial with real coefficients α_i. The goal is to determine the α_i such that the expression

$$\| 1 - zP_n(z) \| \tag{3.2}$$

is minimized on a piecewise linear contour Γ. The contour must be chosen such that Γ is a Jordan curve. Since we presume the matrix to be real Γ will be constructed symmetric with respect to the real axis. Note that the polynomial $1 - zP_n(z)$ is a residual polynomial. Iterative solution algorithms for solving linear systems also construct a residual polynomial. Optimal iterative methods, like GMRES, construct a residual polynomial that is minimal in some sense. As inner product (f, g) of functions f and g we define (see [16, p. 365])

$$(f, g) = \frac{1}{L} \int_{\Gamma} w(z) f(z) \overline{g(z)} \, |\mathrm{d}z|, \tag{3.3}$$

with $L = \int_{\Gamma} w(z) |\mathrm{d}z|$ and $w(z)$ a positive weight function. For the weight function $w(z)$ we choose

$$w(z) = 1. \tag{3.4}$$

This weight function has the advantage that the resulting polynomial preconditioner is less susceptible for bad eigenvalue estimates, i.e., when not all eigenvalues are enclosed by the contour, than when a Chebychev weight function [16] is chosen. The above-defined inner product induces a norm:

$$\| f \|^2 = \frac{1}{L} \int_\Gamma f(z)\overline{f(z)}\,|\mathrm{d}z|. \tag{3.5}$$

Minimizing the residual polynomial $1 - zP_n(z)$ in the sense of least squares in the above defined norm yields the following $n + 1$ equations:

$$\frac{\partial}{\partial \alpha_j} \int_\Gamma (1 - zP_n(z))\overline{(1 - zP_n(z))}\,|\mathrm{d}z| = 0 \quad (j = 0, \dots, n). \tag{3.6}$$

Since we are only considering real matrices, which are matrices with their spectrum symmetric with respect to the real axis, and since $1 - zP_n(z)$ is a polynomial with real coefficients we have that

$$\begin{aligned} &\frac{\partial}{\partial \alpha_j} \int_\Gamma (1 - zP_n(z))\overline{(1 - zP_n(z))}\,|\mathrm{d}z| \\ &\quad = 2\frac{\partial}{\partial \alpha_j} \int_\Gamma (1 - zP_n(z))\overline{(1 - zP_n(z))}\,|\mathrm{d}z| \quad (j = 0, \dots, n). \end{aligned} \tag{3.7}$$

with Γ^+ the nonnegative imaginary part of Γ. We therefore do not have to minimize over the whole contour but only over the nonnegative imaginary part. Suppose that the nonnegative imaginary part of the piecewise linear contour consists of m line pieces and denote the vertices by h_k, $k = 0, \dots, m$. Then (3.7) can be rewritten as

$$\frac{\partial}{\partial \alpha_j} \sum_{k=0}^{m-1} \int_{h_k}^{h_{k+1}} (1 - zP_n(z))\overline{(1 - zP_n(z))}\,|\mathrm{d}z| = 0 \quad (j = 0, \dots, n). \tag{3.8}$$

For the basis polynomials Q_i we simply take

$$Q_i = z^i \quad (i = 0, \dots, n). \tag{3.9}$$

It is a well-known fact that due to the lack of orthogonality between these basis functions, the system (3.8) can become very ill conditioned or even singular. However, since we are only interested in low-degree polynomials we presume that this will not be a problem. A numerically stable choice for the basis functions is described in [12] in which a hybrid GMRES method is proposed. Since the polynomials applied in the second stage of hybrid method are generally of a much higher degree than the degree of the polynomial preconditioner the choice of the basis functions is of much more importance to hybrid methods. With (3.9) for the basis functions we get

$$\begin{aligned} &\sum_{i=0}^{n} \alpha_i \left(\sum_{k=0}^{m-1} \int_{h_k}^{h_{k+1}} z^{j+1}\overline{z^{i+1}} + \overline{z^{j+1}}z^{i+1}\,|\mathrm{d}z| \right) \\ &\quad = \sum_{k=0}^{m-1} \int_{h_k}^{h_{k+1}} z^{j+1} + \overline{z^{j+1}}\,|\mathrm{d}z| \quad (j = 0, \dots, n). \end{aligned} \tag{3.10}$$

The integrals in (3.10) can be computed by parameterization of z on a line piece with a real parameter t.

$$z(t) = \tfrac{1}{2}(h_{k+1} - h_k) \cdot t + \tfrac{1}{2}(h_{k+1} + h_k), \quad -1 \leqslant t \leqslant 1. \tag{3.11}$$

For the integrals we then get

$$\begin{aligned}\int_{h_k}^{h_{k+1}} & z^{j+1}\overline{z^{i+1}} + \overline{z^{j+1}}z^{i+1}\,|\mathrm{d}z| \\ &= \tfrac{1}{2}|h_{k+1} - h_k| \int_{-1}^{1} z(t)^{j+1}\overline{z(t)^{i+1}} + \overline{z(t)^{j+1}}z(t)^{i+1}\,\mathrm{d}t\end{aligned} \tag{3.12}$$

and

$$\int_{h_k}^{h_{k+1}} z^{j+1} + \overline{z^{j+1}}\,|\mathrm{d}z| = \tfrac{1}{2}\,|h_{k+1} - h_k| \int_{-1}^{1} z(t)^{j+1} + \overline{z(t)^{j+1}}\,\mathrm{d}t. \tag{3.13}$$

Note that both t and the integrands are real. The integrals can simply be computed by numerical integration for example by applying a Gaussian quadrature formula. Although the above-described computations seem to be rather complex the actual code can be amazingly simple. In the Appendix, sample FORTRAN code is given for computing the parameters α_i given the vertices h_k.

We have obtained the coefficients of the polynomials preconditioner and now it can be applied in every iteration. There are numerous ways of evaluating the preconditioning polynomial. One can compute, for example, the zeroes of the residual polynomial $1 - zP_n(z)$ and then evaluate the polynomial by a Richardson iteration. However, since the zeroes may be complex, a two-step Richardson formula must be used. A more simple but less stable approach is given by Horner's rule.

```
Horner's rule for evaluating $\boldsymbol{q} = P_n(A)\boldsymbol{p}$.
$\boldsymbol{q} = \alpha_n \boldsymbol{p}$
for $i = n - 1, 0, \ -1$
  $\boldsymbol{q} = A\boldsymbol{q} + \alpha_i \boldsymbol{p}$                                  (3.14)
```

The intermediate results of this method can be so large that information may be lost due to rounding errors. This has never troubled us in practice, however, and we therefore prefer this simple method to Richardson's iteration.

Although the preconditioner described above is based on the maximum modulus lemma, for the polynomial preconditioner it is less dramatic if the contour encloses the origin than for the hybrid methods discussed in the previous section. This is simply due to the fact that, although the polynomial preconditioner might be less effective, the iterative process will always converge, unless the preconditioned matrix is singular.

4. Eigenvalue estimates

For the construction of the polynomial preconditioner eigenvalue estimates are needed. An obvious approach for obtaining eigenvalue estimates is computing the Ritz values, i.e., the eigenvalues of $\tilde{H}_m$, see (2.8). The Ritz values are the zeroes of the residual polynomial generated by the full orthogonalization method (FOM) [6]. GMRES and FOM are closely related methods. GMRES is more robust method since it can never breakdown, except at the exact solution. A root of the FOM residual polynomial, however, can become zero. If that is the case then at that iteration the approximate solution cannot be evaluated with FOM. An elaborate discussion on the relation between the two methods can be found in [18]. In [10] Nachtigal et al. state that the zeroes of the GMRES residual polynomial are often more reliable for their hybrid method and hence are better eigenvalue estimates than the Ritz values. The method they describe for computing these zeroes is to evaluate the residual polynomial explicitly and then to compute the zeroes. This approach is in our experience rather unstable, since the coefficients of the residual polynomial can be very big. This may lead to serious roundoff errors.

The convergence of the zeroes of the GMRES residual polynomial for a symmetric matrix is discussed in [11]. In this paper they denote the zeroes of the residual polynomial by harmonic Ritz values, a terminology we will also adopt. In [11] it is shown that the harmonic Ritz values converge monotonically from above and below to the eigenvalues closest to the origin. This explains the more robust behaviour of GMRES, at least in the symmetric case, since the harmonic Ritz values can never become zero unless the stiffness matrix is singular. It is also proved that the harmonic Ritz values are the eigenvalues of a matrix that only differs by a rank-one update from the tridiagonal Lanczos matrix. This technique for computing the harmonic Ritz values can be generalized in a straightforward way to the nonsymmetric case.

The residual of the FOM algorithm after m iterations is given by

$$r_m = r_0 - \|r_0\| A V_m \tilde{H}_m^{-1} e_1 . \tag{4.1}$$

The residual of the GMRES algorithm is

$$r_m = r_0 - \|r_0\| A V_m H_m^{+} e_m = r_0 - \|r_0\| A V_m (H_m^{\mathrm{T}} H_m)^{-1} \tilde{H}_m^{\mathrm{T}} e_1 \tag{4.2}$$

with H_m^+ the Moore–Penrose inverse of H_m. The Ritz values are the eigenvalues of $\tilde{H}_m$. In [11] it is proved that in the symmetric case the eigenvalues of $\tilde{H}_m^{-\mathrm{T}}(H_m^{\mathrm{T}} H_m)$ are the harmonic Ritz values and from this proof it can be seen that this is true for the nonsymmetric case too. For the matrix $H_m^{\mathrm{T}} H_m$ we can get a more convenient expression

$$H_m^{\mathrm{T}} H_m = \tilde{H}_m^{\mathrm{T}} \tilde{H}_m + h_{m+1,m}^2 e_m e_m^{\mathrm{T}} = \tilde{H}_m^{\mathrm{T}} (I + h_{m+1,m}^2 f_m f_m^{\mathrm{T}}) \tilde{H}_m \tag{4.3}$$

with f_m from the relation

$$\tilde{H}_m^{\mathrm{T}} f_m = e_m \tag{4.4}$$

where $\tilde{H}_m$ must be nonsingular. With these relations we get

$$\tilde{H}_m^{-\mathrm{T}}(H_m^{\mathrm{T}} H_m) = \tilde{H}_m^{-\mathrm{T}} \tilde{H}_m^{\mathrm{T}} (I + h_{m+1,m}^2 f_m f_m^{\mathrm{T}}) \tilde{H}_m = \tilde{H}_m + h_{m+1,m}^2 f_m e_m^{\mathrm{T}} . \tag{4.5}$$

So the harmonic Ritz values can be obtained by computing the eigenvalues of $\tilde{H}_m$, updated with a rank-one update. This technique is stable and therefore preferable to the method described in [10].

5. The construction of a piecewise linear contour

The piecewise linear contour can be constructed according to many different strategies. Which strategy is suitable of course depends on the shape of the spectrum. If the real parts of the eigenvalues are positive, then we propose the following simple strategy:

Suppose we have estimates μ_i for the eigenvalues of the coefficient matrix. The contour is constructed such that it is symmetric with respect to the real axis. For this reason we will only consider μ_i with a nonnegative imaginary part $\mathrm{Im}(\mu_i)$. The goal is to determine suitable vertices. An eigenvalue estimate μ_k is selected to be a vertex if it fulfils one of the two following conditions:

1. μ_k has a greater imaginary part than all other μ_i with smaller or equal real parts $\mathrm{Re}(\mu_i)$, hence it has the property

$$\mathrm{Re}(\mu_i) \leqslant \mathrm{Re}(\mu_k) \Rightarrow \mathrm{Im}(\mu_i) < \mathrm{Im}(\mu_k) \vee \mu_i = \mu_k \quad \forall \mu_i, \tag{5.1}$$

2. μ_k has a greater imaginary part than all other μ_i with greater or equal real parts $\mathrm{Re}(\mu_i)$, hence it has the property

$$\mathrm{Re}(\mu_i) \geqslant \mathrm{Re}(\mu_k) \Rightarrow \mathrm{Im}(\mu_i) < \mathrm{Im}(\mu_k) \vee \mu_i = \mu_k \quad \forall \mu_i, \tag{5.2}$$

We are only determining vertices for the nonnegative imaginary part of the contour. Therefore, the first and last vertex must be real. If none of the μ_i is real then these first and last vertices must be added to the selected eigenvalue estimates. Finally, the vertices must be ordered along Γ^+, so that Γ is a Jordan curve.

The strategy described above is unsuited for properly approximating a "boomerang"-shaped spectrum. Moreover, if the coefficient matrix does not have a positive definite symmetric part, then a contour constructed according to the above-described strategy will always enclose the origin, also when the spectrum is similar to that of Fig. 1. It may therefore sometimes be necessary to also select eigenvalue estimates to be vertices if they have one of two supplementary properties:

3. μ_k has a smaller imaginary part than all other μ_i with smaller or equal real parts $\mathrm{Re}(\mu_i)$, hence it has the property

$$\mathrm{Re}(\mu_i) \leqslant \mathrm{Re}(\mu_k) \Rightarrow \mathrm{Im}(\mu_i) > \mathrm{Im}(\mu_k) \vee \mu_i = \mu_k \quad \forall \mu_i, \tag{5.3}$$

4. μ_k has a smaller imaginary part than all other μ_i with greater or equal real parts $\mathrm{Re}(\mu_i)$, hence it has the property

$$\mathrm{Re}(\mu_i) \geqslant \mathrm{Re}(\mu_k) \Rightarrow \mathrm{Im}(\mu_i) > \mathrm{Im}(\mu_k) \vee \mu_i = \mu_k \quad \forall \mu_i. \tag{5.4}$$

Again, the first and last vertex must be real and the vertices must be ordered along Γ^+, so that Γ is a Jordan curve.

6. Numerical experiments

In this section some numerical experiments will be described. The goals of these experiments are:

- to examine the effectiveness of the polynomial preconditioner for GMRES(m) and full GMRES,
- and to compare the effect of the two strategies for obtaining eigenvalue estimates.

The computations described in this section are performed using the finite element method program DIANA [3] on a Silicon Graphics Iris workstation in double precision arithmetic.

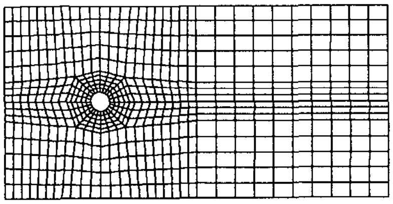

Fig. 2. Element mesh of the test problem.

The test problem we have chosen is described in [19, pp. 44–47]. It models vortex shedding behind a circular cylinder. The finite element formulation is based on the work of Hansbo [7]. The mesh is given in Fig. 2.

The model consists of 596 quadratic 8 noded elements. Each node has two velocity degrees of freedom and one pressure degree of freedom. The test problem is both time dependent and nonlinear by nature. During the nonlinear solution phase, systems of linear equations have to be solved. We have selected one typical linear system for our experiments. This system is nonsymmetric and consists of 5306 equations.

The piecewise linear contours are constructed using (5.1) and (5.2) only. This is because it was a priori known that the spectrum of the matrix contains only eigenvalues with a positive real part. The matrix–vector products in the GMRES algorithm and in the polynomial preconditioner are performed element-by-element. The polynomial preconditioner is combined with an element-by-element preconditioning technique, i.e., we have (implicity) multiplied the linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{6.1}$$

by a preconditioning matrix $\boldsymbol{P}^{-1}$

$$\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{P}^{-1}\boldsymbol{b} \tag{6.2}$$

and hence the polynomial preconditioner is a polynomial in $\boldsymbol{P}^{-1}\boldsymbol{A}$. The element-by-element preconditioner we have applied is the rowsum-EBE preconditioner which is discussed in [20]. The iterative process was stopped if the residual was reduced by a factor 10^6.

For most practical problems GMRES will have to be restarted after a number of iterations. In the experiments we will describe first, GMRES was restarted every 20 iterations. The parameters for the polynomial preconditioner are computed after the first cycle of iterations. The parameters are computed from the Ritz values. Fig. 3 shows the Ritz values and the piecewise linear contour.

The Ritz value closest to the origin is approximately 0.025. Fig. 4 shows the reduction of the norm of the scaled residual as a function of the number of matrix–vector multiplications.

The number of matrix–vector multiplications is a measure for the number of operations needed for reducing the norm of the residual sufficiently. From Fig. 4 it is apparent that by applying the polynomial preconditioner an important reduction in the number of matrix–vector multiplications can be gained. Applying a polynomial preconditioner of degree 1 already gives a considerable improvement, no significant gain is obtained if the degree of the polynomial is taken to be higher than 2. In Table 1 the CPU-times for the complete solution stage are tabulated, so the construction of the rowsum-EBE preconditioner and the initial 20 iterations are also included.

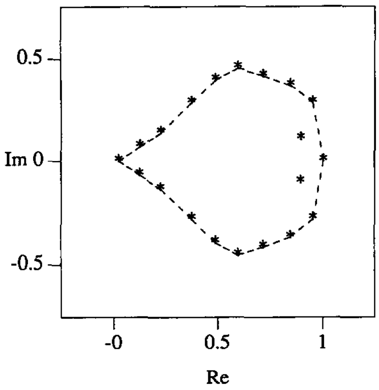

Fig. 3. Ritz values and contour after 20 iterations.

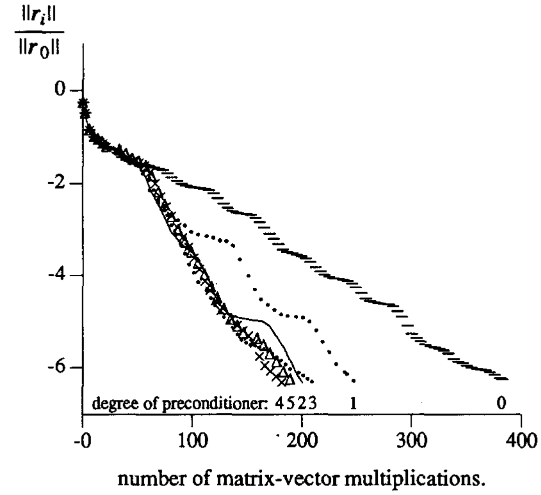

Fig. 4. The norm of the residual as a function of the number of matrix–vector multiplications.

Table 1
CPU-times of GMRES(20) with polynomial preconditioner based on Ritz values

Degree of preconditioner	Iterations	CPU-time in s
0	363	163.7
1	127	101.9
2	77	81.9
3	64	83.4
4	50	72.4
5	46	75.3

The CPU-time is reduced by approximately a factor 2 when a polynomial preconditioner of degree 2 or higher is applied.

The next experiment is meant to examine the effect of using the zeroes of the GMRES polynomial instead of Ritz values for computing the polynomial preconditioner. The zeroes and contour after 20 iterations are shown in Fig. 5.

The zero closest to the origin is 0.069 + 0.053i. The reduction of the norm of the residual as a function of the number of matrix–vector multiplications is shown in Fig. 6.

Fig. 6 does not show significantly higher rates of convergence for the methods with a polynomial preconditioner based on harmonic Ritz values, compared with the results shown in Fig. 4. The CPU-times are tabulated in Table 2.

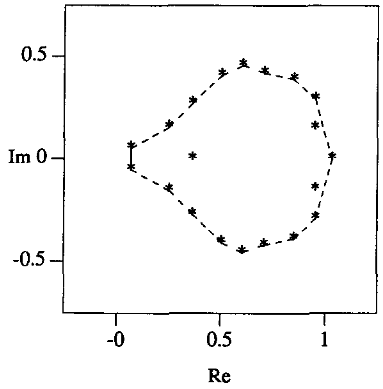

Fig. 5. Harmonic Ritz values and contour after 20 iterations.

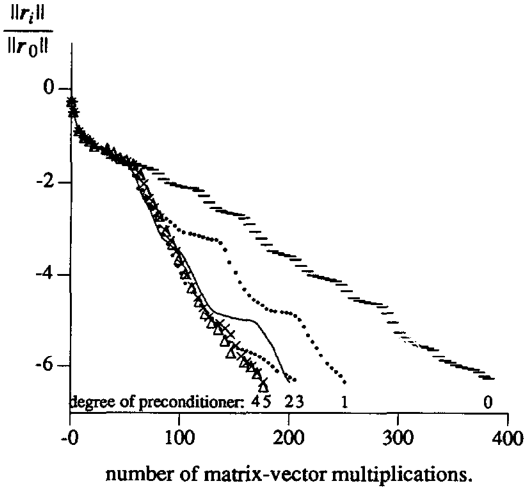

Fig. 6. The norm of the residual as a function of the number of matrix–vector multiplications.

The above results are of course insufficient for drawing soundly based conclusions. From our experience we tend to believe that one cannot expect that the harmonic Ritz values yield a better polynomial preconditioner, at least if the coefficient matrix has a positive-definite symmetric part. We hope to address this topic more thoroughly in a future work.

Now we will describe an experiment in which the polynomial preconditioner is combined with full GMRES. For this experiment, the polynomial preconditioners are based on Ritz values. Full GMRES minimizes the residual norm over the Krylov subspace (2.1). We can therefore not expect that by applying the polynomial preconditioner the number of matrix–vector products can be reduced. This is confirmed by the results for the convergence behaviour given in Fig. 7.

The orthogonalization of the basis for the Krylov subspace becomes increasingly expensive in every iteration. Therefore, although the number of matrix–vector products will increase, the number of floating points operations may decrease considerably if a polynomial preconditioner is applied. The CPU-times given in Table 3 confirm this.

Table 2
CPU-times of GMRES(20) with polynomial preconditioner based on harmonic Ritz values

Degree of preconditioner	Iterations	CPU-time in s
0	363	163.7
1	129	103.2
2	77	82.0
3	63	82.5
4	49	70.5
5	44	71.0

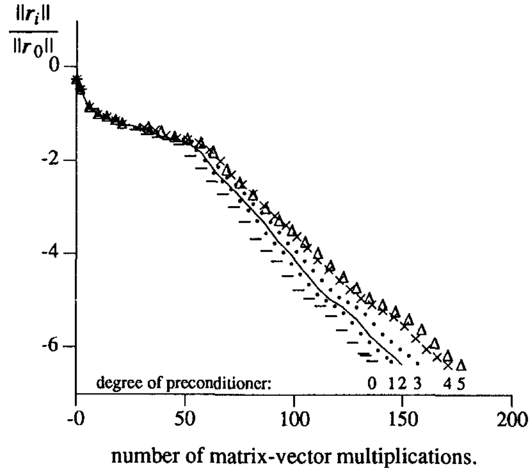

Fig. 7. The norm of the residual as a function of the number of matrix–vector multiplications.

Table 3
CPU-times of full GMRES with polynomial preconditioner based on Ritz values

Degree of preconditioner	Iterations	CPU-time in s
0	135	95.7
1	20 + 61	69.4
2	20 + 42	66.4
3	20 + 33	67.6
4	20 + 29	72.3
5	20 + 24	71.3

Note that in the above experiment GMRES had to be restarted after the twentieth iteration if a polynomial preconditioner was applied. GMRES without polynomial preconditioner need not be, and was not restarted. Therefore, at the end of the process 135 basis vectors of the Krylov subspace had to be stored. For example if a preconditioner of degree 1 was applied, 61 basis vectors had to be stored and only 24 vectors if a preconditioner of degree 5 was applied!

7. Summary and conclusions

In this paper a least-squares polynomial preconditioner of low degree for use with the GMRES algorithm is discussed. The parameters of the polynomial preconditioner are computed from eigenvalue estimates that can be obtained from the GMRES iterative process. Since we have considered polynomials of low degree only stability is not a major problem. The algorithm is simple and therefore will applicable in real life.

Two different methods for obtaining eigenvalue estimates are discussed. A simple and stable approach is to calculate the Ritz values with Arnoldi's method. In literature it was reported that the zeroes of the GMRES residual polynomial may yield more robust eigenvalue estimates. We describe in this paper a stable strategy for computing these zeroes. An experiment where polynomial preconditioners based on these two eigenvalue estimates are compared renders insufficient information to draw soundly based conclusions, and more research on this topic will be necessary.

Experiments indicate that a considerable reduction of the CPU-time can be reached if the polynomial preconditioner is applied in combination with GMRES with restarts after cycles of iterations. The experiments do not indicate that a significantly greater reduction of the CPU-time can be achieved if a polynomial preconditioner of high degree is applied. We observed only a marginal reduction of the CPU-time if a polynomial preconditioner of degree higher than two was applied. For full GMRES no reduction of the number of matrix–vector products can be achieved. However, since the orthogonalization of the basis for the Krylov subspace becomes increasingly expensive an important reduction of the CPU-time can be achieved by applying the polynomial preconditioner. Moreover, since the number of iterations is decreased if the polynomial preconditioner is applied, far less basis vectors for the Krylov subspace need to be stored. We have shown an example where by applying a polynomial preconditioner in combination with full GMRES both the CPU-time and the storage requirements were reduced significantly.

The success of the polynomial preconditioner depends, especially with full GMRES, to a great extent on how expensive a matrix–vector multiplication is with respect to the orthogonalization process for the basis of the Krylov subspace. If the matrix–vector product can be performed cheaply, for example because the matrix has a nice diagonal structure with few diagonals with nonzero elements, then we can expect a substantial reduction of the CPU-time if the polynomial preconditioner is applied. The matrix–vector multiplications in the experiments described in this paper were performed element-by-element. Therefore, in these experiments, the matrix–vector multiplication was an expensive operation, but even then the CPU-time was reduced significantly.

The polynomial preconditioner can also be combined with methods other than GMRES. With methods such as CG, CGS and bi-CGSTAB, eigenvalue estimates can also be obtained. We cannot expect a reduction of the CPU-time if the polynomial preconditioner is applied with CG, since CG is, like full GMRES, optimal in some norm and we cannot reduce the number of matrix–vector products. CGS and bi-CGSTAB are not optimal methods. Whether it is worthwhile to combine the polynomial preconditioner with one of these methods is an open question.

Acknowledgements

The author would like to thank Dr. C. Vuik for his advice on many aspects of this work and Prof. Dr. H.A. van der Vorst for his instructive suggestions.

Appendix. Sample FORTRAN code for computing the parameters of the polynomial preconditioner

```
         SUBROUTINE PARMTR( DEGREE, ALPHA, M, VERTEX, SINGLR )
C
C........................................................................
C...   INPUT:       DEGREE: DEGREE OF THE POLYNOMIAL PRECONDITIONER
C...                M:      NUMBER OF VERTICES
C...                VERTEX: VERTICES
C...
C...   OUTPUT:      ALPHA:  COEFFICIENTS OF POLYNOMIAL PRECONDITIONER
C...                SINGLR: SINGULAR PARAMETER SYSTEM
C...
C...   PURPOSE:     TO COMPUTE ALPHA
C...
C...   LINPACK ROUTINES CALLED: DSICO, DSISL
C...
C...   PROGRAMMED BY MARTIN VAN GIJZEN.   TNO-BOUW  141292
C........................................................................
C
      DOUBLE PRECISION ALPHA(*)
      COMPLEX          VERTEX(*)
      INTEGER          DEGREE, M
      LOGICAL          SINGLR
C
      REAL             W, F, G
      INTEGER          I, J, K, L
      COMPLEX          Z, C1, C2
```

```
C
C...     SPACE FOR SYSTEM TO BE SOLVED:
      DOUBLE PRECISION A(11,11), WORK(11), RCOND
      INTEGER          IPVT(11)
C
C...     GAUSS POINTS AND GAUSS WEIGHTS FOR NUMERICAL INTEGRATION:
      REAL             GAUSS(2,6)
      DATA GAUSS/ -0.93246951,               0.17132449,
     2            -0.66120938,               0.36076157,
     3            -0.23861918,               0.46791393,
     4            +0.23861918,               0.46791393,
     5            +0.66120938,               0.36076157,
     6            +0.93246951,               0.17132449 /
C
      DATA IPVT/ 11 * 0 /, A/ 121 * 0.D0 /, WORK/ 11 * 0.D0 /
C
C...     INTEGRANDS OF INTEGRALS TO BE EVALUATED:
      F(Z,I,J) = REAL( Z**(J+1) * CONJG( Z**(I+1) ) )
      G(Z,J)   = REAL( Z**(J+1) )
C
      DO 10, I = 1, DEGREE+1
         ALPHA(I) = 0.D0
   10 CONTINUE
C
C...     DO FOR ALL LINEPIECES
      DO 50, K = 1, M-1
C
C...        COEFFICIENTS FOR PARAMETERIZATION:
         C1 = 0.5 * ( VERTEX(K+1) - VERTEX(K) )

         C2 = 0.5 * ( VERTEX(K+1) + VERTEX(K) )
C
C...        GAUSS INTEGRATION, 6 GAUSS POINTS
         DO 40, L = 1, 6
            Z = C1*GAUSS(1,L) + C2
            W = ABS( C1 ) * GAUSS(2,L)
C
C...           ADD CONTRIBUTION OF LINEPIECE TO SYSTEM
            DO 30, J = 0, DEGREE
               DO 20, I = 0, J
                  A(I+1,J+1) = A(I+1,J+1) + W*F(Z,I,J)
   20          CONTINUE
               ALPHA(J+1) = ALPHA(J+1) + W*G(Z,J)
   30       CONTINUE
   40    CONTINUE
   50 CONTINUE
C
C...     SOLVE SYSTEM TO DETERMINE PARAMETERS OF POLYNOMIAL
      CALL DSICO( A, 11, DEGREE+1, IPVT, RCOND, WORK )
      SINGLR = 1.D0+RCOND .EQ. 1.D0
      IF ( .NOT. SINGLR ) CALL DSISL( A, 11, DEGREE+1, IPVT, ALPHA )
C
      END
```

References

[1] T.M. Apostol, *Mathematical Analysis* (Addison-Wesley, Reading, MA, 1957) 537.

[2] W.E. Arnoldi, The principle of minimized iteration in the solution of the matrix eigenvalue problem, *Quart. Appl. Math.* **9** (1951) 17–29.

[3] R. de Borst, G.M.A. Kusters, P. Nauta and F.C. de Witte, DIANA – a comprehensive, but flexible finite element system, in: C.A. Brebbia, Ed., *Finite Element Systems: A Handbook* (Springer, Berlin, 1985).
[4] H. Elman, Y. Saad and P.E. Saylor, A hybrid Chebychev Krylov subspace algorithm for solving nonsymmetric systems of linear equations, *SIAM J. Sci. Statist. Comput.* **7** (1986) 840–855.
[5] R. Fletcher, *Conjugate Gradient Methods for Indefinite Systems*, Lecture Notes Math. Vol. 506 (Springer, Berlin, 1976) 73–89.
[6] G.H. Golub and C.F. van Loan, *Matrix Computations* (North Oxford Academic, Oxford, 1983, 1989).
[7] P. Hansbo, Adaptive and streamline diffusion procedures in the finite element method, Dissertation, Dept. of Structural Mechanics, Chalmers Univ. of Technology, Göteborg, 1989.
[8] M.R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards* **49** (6) (1952) 409–436.
[9] T.A. Manteuffel, The Tchebychev iteration for nonsymmetric linear systems, *Numer. Math.* **28** (1977) 307–327.
[10] N.M. Nachtigal, L. Reichel and L.N. Trefethen, A hybrid GMRES algorithm for nonsymmetric linear systems, *SIAM J. Matrix Anal. Appl.* **13** (3) (1992) 796–825.
[11] C. Paige, B.N. Parlett and H.A. van der Vorst, Approximate solutions and eigenvalue bounds from Krylov subspaces, PAM-579, Center for Pure and Applied Mathematics, University of California, Berkeley, 1993.
[12] Y. Saad, Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems, *SIAM J. Numer. Anal.* **24** (1) (1987) 155–169.
[13] Y. Saad and M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **7** (3) (1986) 856–869.
[14] D.C. Smolarski and P. Saylor, An optimum iterative method for solving any linear system with a square matrix, *BIT* **28** (1988) 163–178.
[15] P. Sonneveld, CGS: a fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **10** (1989) 36–52.
[16] G. Szegő, *Orthogonal Polynomials* (American Mathematical Society, Providence, RI, 1939, 2nd ed., 1975).
[17] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **13** (2) (1992) 631–644.
[18] H.A. van der Vorst and C. Vuik, The superlinear convergence behaviour of GMRES, *J. Comput. Appl. Math.* **48** (3) (1993) 327–341.
[19] F.N. van de Vosse, Numerical analysis of carotid artery flow, Dissertation, Eindhoven University of Technology, 1987.
[20] M.B. van Gijzen, An analysis of element-by-element preconditioners for nonsymmetric problems, *Comput. Methods Appl. Mech. Engng.* **105** (1993) 23–40.