IMA Journal of Numerical Analysis (1994) **14,** 563–581

A Newton basis GMRES implementation

Z. Bai†
Department of Mathematics, University of Kentucky, Lexington, KY 40506, USA

D. Hu†
Center for Research on Parallel Computation, Rice University, Houston, TX 77251, USA

AND

L. Reichel†‡
Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242, USA

[Received 17 April 1991 and in revised form 20 April 1994]

The GMRES method by Saad and Schultz is one of the most popular iterative methods for the solution of large sparse non-symmetric linear systems of equations. The implementation proposed by Saad and Schultz uses the Arnoldi process and the modified Gram–Schmidt (MGS) method to compute orthonormal bases of certain Krylov subspaces. The MGS method requires many vector–vector operations, which can be difficult to implement efficiently on vector and parallel computers due to the low granularity of these operations. We present a new implementation of the GMRES method in which, for each Krylov subspace used, we first determine a Newton basis, and then orthogonalize it by computing a QR factorization of the matrix whose columns are the vectors of the Newton basis. In this way we replace the vector–vector operations of the MGS method by the task of computing a QR factorization of a dense matrix. This makes the implementation more flexible, and provides a possibility to adapt the computations to the computer at hand in order to achieve better performance.

1. Introduction

The generalized minimal residual (GMRES) method introduced by Saad & Schultz (Saad & Schultz (1986), Saad (1989)) is one of the most powerful iterative schemes for the numerical solution of large sparse non-symmetric linear systems of equations

$$Ax = b \tag{1.1}$$

† E-mail addresses: bai@ms.uky.edu dyhu@rice.edu reichel@mcs.kent.edu

‡ Research supported in part by the IBM Bergen Scientific Center, the Center for Research on Parallel Computation at Rice University, a National Research Council fellowship and NSF Grant DMS-9002884.

where the matrix $A \in \mathbb{R}^{N\times N}$ is non-singular and the right hand side $b \in \mathbb{R}^N$. In this method one chooses an initial approximate solution x_0, defines the residual vector $r_0 := b - Ax_0$, and then computes a better approximate solution $x_1 := x_0 + z_0 \in x_0 + \mathbf{K}_m(A, r_0)$, such that

$$\|b - Ax_1\| = \min_{z \in \mathbf{K}_m(A,r_0)} \|b - A(x_0 + z)\| \tag{1.2}$$

where

$$\mathbf{K}_m(A, r_0) := \text{span}\{r_0, Ar_0, ..., A^{m-1}r_0\} \tag{1.3}$$

is a Krylov subspace and m is a given positive integer. Throughout this paper $\|\cdot\|$ denotes the Euclidean vector norm or the corresponding induced matrix norm.

Saad & Schultz (1986) solve the minimization problem (1.2) by first computing an orthonormal basis $\{v_k\}_{k=1}^{m+1}$ of $\mathbf{K}_{m+1}(A, r_0)$ with $v_1 := r_0/\|r_0\|$ by the Arnoldi process. Define the matrices $V_m := [v_1, v_2, ..., v_m]$ and $V_{m+1} := [v_1, v_2, ..., v_{m+1}]$. The Arnoldi process also yields an essentially upper Hessenberg matrix $H := [\eta_{jk}] \in \mathbb{R}^{(m+1)\times m}$, such that

$$AV_m = V_{m+1}H. \tag{1.4}$$

Substitution of (1.4) into (1.2) yields

$$\begin{aligned}\min_{z \in \mathbf{K}_m(A,r_0)} \|r_0 - Az\| &= \min_{y \in \mathbb{R}^m} \|r_0 - AV_m y\| \\ &= \min_{y \in \mathbb{R}^m} \|r_0 - V_{m+1}Hy\| \\ &= \min_{y \in \mathbb{R}^m} \| \|r_0\| e_1 - Hy\| \end{aligned} \tag{1.5}$$

where we have used that $V_{m+1}^{\mathrm{T}} r_0 = \|r_0\| e_1$. Throughout this paper $e_j = [0, ..., 0, 1, 0, ..., 0]^{\mathrm{T}}$ denotes the jth axis vector of appropriate dimension. The least-squares problem (1.5) is solved for $y_0 \in \mathbb{R}^m$ by QR factorization of H, and we obtain the new approximate solution x_1 of (1.1) from $x_1 := x_0 + V_m y_0$. For reasons of numerical stability, Saad & Schultz (1986) implement the Arnoldi process using the modified Gram–Schmidt (MGS) method. This leads to the following algorithm.

ALGORITHM 1.1 (GMRES implementation by Saad & Schultz (1986))

Input: m, x_0, $r_0 := b - Ax_0$;

Output: x_1, H

Arnoldi process:†

$v_1 := r_0/\|r_0\|$;

for $k := 1, 2, \ldots, m$ do

$w := Av_k$;

for $j := 1, 2, \ldots, k$ do

$$\eta_{jk} := w^{\mathrm{T}} v_j; \qquad w := w - \eta_{jk} v_j; \tag{1.6}$$

end j;

$$\eta_{k+1,k} := \|w\|; \qquad v_{k+1} := w/\eta_{k+1,k}; \tag{1.7}$$

end k;

Solve (1.5) for y_0 by computing the QR factorization of H using m Givens rotations; $x_1 := x_0 + V_m y_0$, where $V_m = [v_1, v_2, \ldots, v_m]$. □

For future reference, we remark that in the *Arnoldi method* for computing approximations of eigenvalues of A, one determines the spectrum of the $m \times m$ upper Hessenberg matrix

$$H_m := V_m^{\mathrm{T}} A V_m. \tag{1.8}$$

Note that H_m consists of the first m rows of the matrix $H \in \mathbb{R}^{(m+1)\times m}$ determined by Algorithm 1.1.

The storage requirement of Algorithm 1.1 grows linearly with m, and the number of arithmetic operations required grows quadratically with m. Therefore, one generally chooses a fixed value of m, say $10 \leq m \leq 50$, and computes an approximate solution x_1 by Algorithm 1.1. If the norm of the residual error $r_1 := b - Ax_1$ is not sufficiently small, then one seeks to improve x_1 by solving a minimization problem analogous to (1.2). This gives rise to the cyclic GMRES algorithm, also introduced by Saad & Schultz (1986).

ALGORITHM 1.2 (Cyclic GMRES(m) Algorithm (Saad & Schultz (1986))
Input: m, x_0, $r_0 := b - Ax_0$, $\varepsilon > 0$;

Output: approximate solution x_l such that $\|b - Ax_l\| \leq \varepsilon$;

for $j := 0, 1, 2, \ldots$ until $\|r_j\| \leq \varepsilon$ do

$$z_j := \arg \min_{z \in \mathbf{K}_m(A, r_j)} \|r_j - Az\| \tag{1.9}$$

$$x_{j+1} := x_j + z_j; \tag{1.10}$$

$r_{j+1} := r_j - Az_j$;

end j. □

The solution of (1.2) or (1.9) by the Arnoldi process gives rise to many

† The elements η_{jk} defined in the algorithm are the non-vanishing entries of the matrix $H \in \mathbb{R}^{(m+1)\times m}$ in (1.4).

vector–vector operations; see (1.6). These operations can be difficult to implement efficiently on vector and parallel computers due to their low granularity; see for example Dongarra *et al* (1986, 1991), Chronopoulos & Kim (1990). The present paper describes a scheme for the solution of (1.9) in which the MGS method in Algorithm 1.1, i.e., the vector–vector operations (1.6), is replaced by the QR factorization of a dense $N \times (m+1)$ matrix. This has the advantage of making the implementation of our scheme more flexible than the implementation of Algorithm 1.1; the QR factorization can be computed by an algorithm that is well suited for the computer at hand. The number of arithmetic operations required by our implementation of the cyclic GMRES(m) algorithm is roughly the same as if Algorithm 1.1 were used for the solution of (1.9).

Other approaches to avoid the vector–vector operations (1.6) are described by Hindmarsh & Walker (1986) and Walker (1988, 1989). In Walker (1988, 1989) two schemes that use Householder transformations, instead of the MGS method (1.6) and (1.7), for computing orthonormal bases of Krylov subspaces $\mathbf{K}_{m+1}(A, r_j)$ are presented. These schemes might perform well on certain computers. However, they require roughly two to three times more arithmetic operations (depending on implementation details) than Algorithm 1.1, and this can make these schemes slower than Algorithm 1.1.

In Hindmarsh & Walker (1986) the Krylov subspaces are represented by using scaled monomial bases in the matrix A, i.e.,

$$\mathbf{K}_{m+1}(A, r_j) = \text{span}\,\{\sigma_0 r_j, \sigma_1 A r_j, \ldots, \sigma_m A^m r_j\}$$

where the $\sigma_k > 0$ are scaling factors. The QR factorization of $N \times (m+1)$ matrices with columns $\sigma_k A^k r_j$, $0 \leq k \leq m$, are computed by using Householder transformations. Computed examples in Hindmarsh & Walker (1986) and Walker (1989) show this approach to yield poor convergence or even no convergence. This is due to the fact that the bases $\{\sigma_k A^k r_j\}_{k=0}^m$ can be severely ill-conditioned; see also Section 2 for a discussion. Our implementation is a modification of this scheme. We replace the scaled monomial bases by better conditioned bases of Newton form. This change of bases eliminates the numerical difficulties encountered with the monomial bases.

We introduce notation necessary to describe our implementation. Let Π_{m-1} denote the set of polynomials of degree at most $m-1$. The minimization problem (1.9) is equivalent to the problem

$$\min_{p \in \Pi_{m-1}} \|r_j - Ap(A)r_j\| \tag{1.9'}$$

whose solution we denote by p_j. Then (1.10) can be written as

$$x_{j+1} := x_j + p_j(A)r_j. \tag{1.10'}$$

We represent $p \in \Pi_{m-1}$ by bases of Newton form $\{\sigma_k \Pi_{l=1}^k (\cdot - \lambda_l)\}_{k=0}^{m-1}$, where the $\sigma_k > 0$ are scaling factors and the parameters $\lambda_l \in C$ are chosen with the aim of making these bases fairly well-conditioned. The choice of the λ_l is discussed in detail in Section 2. We propose to let the λ_l be suitably ordered eigenvalues of the Hessenberg matrix H_m defined by (1.8) and computed by Algorithm 1.1. We remark that the matrix H_m and its spectrum only have to be computed once.

Introduce the matrix

$$B_m := \left[\sigma_0 r_j,\ \sigma_1 (A - \lambda_1 I) r_j, \ldots, \sigma_{m-1} \prod_{l=1}^{m-1} (A - \lambda_l I) r_j \right]. \tag{1.11}$$

Then for any $p \in \Pi_{m-1}$ we have $p(A) r_j = B_m d$ for some vector $d \in \mathbb{C}^m$. The minimization problem (1.9′) is equivalent to the problem

$$\min_{d \in \mathbb{C}^m} \| r_j - A B_m d \| \tag{1.9″}$$

whose solution we denote by d_j. Then (1.10′) becomes

$$x_{j+1} := x_j + B_m d_j. \tag{1.10″}$$

We are now in a position to outline our implementation of the cyclic GMRES(m) algorithm: given an initial approximate solution x_0, (i) choose m and compute x_1, as well as the $m \times m$ Hessenberg matrix H_m by Algorithm 1.1; (ii) determine the eigenvalues of H_m and order them for numerical stability; and (iii) solve (1.9) for $j \geqslant 1$ by solving the equivalent problem (1.9″) and updating the approximate solution x_j of (1.1) by (1.10″). We solve (1.9″) by computing a QR factorization of the matrix AB_m. Details of this implementation are presented in Section 3. Computed examples are described in Section 4, and Section 5 contains a summary and a brief discussion on possible extensions.

2. A Krylov subspace Newton basis

If the matrix AB_m in (1.9″) is severely ill-conditioned, then the corrections $B_m d_j$ to x_j in (1.10″) cannot be computed with sufficient accuracy. The vector $B_m d_j$ may be less sensitive to ill-conditioning of AB_m than the least-squares solution d_j of (1.9″) is, but nonetheless, extreme ill-conditioning of AB_m gives rise to poor accuracy in the vector $B_m d_j$. In the algorithm presented in Section 3 we determine the QR factorization of AB_m by first computing the QR factorization of

$$B_{m+1} := \left[\sigma_0 r_j,\ \sigma_1 (A - \lambda_1 I) r_j, \ldots, \sigma_m \prod_{l=1}^{m} (A - \lambda_l I) r_j \right]. \tag{2.1}$$

This section shows how the condition number

$$\kappa(B_{m+1}) := \frac{\max_{\|c\|=1} \| B_{m+1} c \|}{\min_{\|c\|=1} \| B_{m+1} c \|} \tag{2.2}$$

where $c = [\gamma_0, \gamma_1, \ldots, \gamma_m]^T \in \mathbb{C}^{m+1}$, can be related to the condition number of polynomial bases. Results by Gautschi (1979, 1984) and Reichel (1985) on the conditioning of monomial bases, as well as on the conditioning of polynomial bases in Newton form in Fischer & Reichel (1989) and Reichel (1990), are suggestive for the choice of basis of $\mathbf{K}_{m+1}(A, r_j)$. It is the aim of this section to provide a heuristic motivation for a basis of Newton form for $\mathbf{K}_{m+1}(A, r_j)$.

For simplicity we assume that the matrix A is normal and has distinct eigenvalues. Let $A = U \Lambda U^*$ be a spectral decomposition with U unitary and Λ

diagonal. Here and below * denotes transposition and complex conjugation. Let $\lambda(A)$ denote the spectrum and $\rho = \rho(A)$ the spectral radius of A. We first study the condition number $\kappa(B_{m+1})$ when all the λ_k in (2.1) vanish. To simplify the analysis we assume that the vector r_j in (2.1) is given by $r_j = Ue$, with $e := N^{-1/2}[1, 1, \ldots, 1]^T$, and we choose the scaling factors $\sigma_k := \rho^k$. Then the lengths of the columns $\sigma_k A^k r_j$ of B_{m+1} are bounded by $1 \leq \|\sigma_k A^k r_j\| \leq N^{1/2}$. We obtain the inequalities

$$\begin{aligned}\max_{\|c\|=1} \|B_{m+1}c\| &= \max_{\|c\|=1} \|U^*B_{m+1}c\| \geq (m+1)^{-1/2} \max_{\|c\|_\infty=1} \|U^*B_{m+1}c\|_\infty \\ &= (m+1)^{-1/2} \max_{\|c\|_\infty=1} \left\| \sum_{k=0}^{m} \gamma_k \sigma_k A^k e \right\|_\infty \\ &= (m+1)^{-1/2} N^{-1/2} \max_{\|c\|_\infty=1} \max_{\lambda \in \lambda(A)} \left| \sum_{k=0}^{m} \gamma_k \left(\frac{\lambda}{\rho}\right)^k \right| \end{aligned} \tag{2.3}$$

and

$$\begin{aligned}\min_{\|c\|=1} \|B_{m+1}c\| &= \min_{\|c\|=1} \|U^*B_{m+1}c\| \leq N^{1/2} \min_{\|c\|_\infty=1} \|U^*B_{m+1}c\|_\infty \\ &= N^{1/2} \min_{\|c\|_\infty=1} \left\| \sum_{k=0}^{m} \gamma_k \sigma_k A^k e \right\|_\infty = \min_{\|c\|_\infty=1} \max_{\lambda \in \lambda(A)} \left| \sum_{k=0}^{m} \gamma_k \left(\frac{\lambda}{\rho}\right)^k \right| \end{aligned} \tag{2.4}$$

where $\|\cdot\|_\infty$ denotes the uniform norm.

Formulas (2.3) and (2.4) enable us to relate $\kappa(B_{m+1})$ to the condition number of a scaled monomial basis of Π_m. Following Gautschi (1979, 1984), we let $\{\phi_j\}_{j=0}^m$ be a basis of Π_m defined on a given compact set $S \subset \mathbb{C}$, and introduce the operator $P : \mathbb{C}^{m+1} \to \Pi_m$,

$$(Pc)(\zeta) := \sum_{k=0}^{m} \gamma_k \phi_k(\zeta), \qquad \zeta \in S.$$

We equip the domain of P with the norm $\|\cdot\|_\infty$ and the range with the norm $\|f\|_S := \max_{z \in S} |f(z)|$. Let P^{-1} denote the inverse of P. We define the condition number of P by

$$\tilde{\kappa}_S(\{\phi_j\}_{j=0}^m) = \tilde{\kappa}(P) := \|P\| \, \|P^{-1}\|,$$

where the norms on the right are induced operator norms. Thus,

$$\tilde{\kappa}_S(\{\phi_j\}_{j=0}^m) = \max_{\|c\|_\infty=1} \max_{\zeta \in S} \left| \sum_{k=0}^{m} \gamma_k \phi_k(\zeta) \right| \Big/ \min_{\|c\|_\infty=1} \max_{\zeta \in S} \left| \sum_{k=0}^{m} \gamma_k \phi_k(\zeta) \right|. \tag{2.5}$$

Let $\phi_j(\lambda) := (\lambda/\rho)^j$, $0 \leq j \leq m$, and $S := \lambda(A)$. Then formulas (2.2)–(2.5) show that there is a vector r_j of unit length, such that

$$\kappa(B_{m+1}) \geq (m+1)^{-1/2} N^{-1/2} \tilde{\kappa}_{\lambda(A)}\left(\left\{ \left(\frac{\lambda}{\rho}\right) \right\}_{j=0}^{m} \right). \tag{2.6}$$

Gautschi (1979) has shown that if S is the interval $[-\rho, \rho]$, then

$$\tilde{\kappa}_S\left(\left\{ \left(\frac{\lambda}{\rho}\right) \right\}_{j=0}^{m} \right)$$

grows as $(1+\sqrt{2})^m$ with m. The case when S is an ellipse is discussed in Reichel (1985). The condition number

$$\tilde{\kappa}_S\left(\left\{\left(\frac{\lambda}{\rho}\right)\right\}_{j=0}^{m}\right)$$

grows exponentially with m for many compact sets $S \subset \mathbb{C}$, and in view of (2.6) this suggests that $\kappa(B_{m+1})$ grows exponentially with m (for $m \leq N$) for many matrices A and certain vectors r_j.

We now determine an upper bound for $\kappa(B_{m+1})$, using the inequalities

$$\max_{\|c\|=1} \|B_{m+1}c\| \leq N^{1/2} \max_{\|c\|_\infty=1} \|U^*B_{m+1}c\|_\infty = N^{1/2} \max_{\|c\|_\infty=1} \left\| \sum_{k=0}^{m} \gamma_k \sigma_k \Lambda^k e \right\|_\infty$$

$$= \max_{\|c\|_\infty=1} \max_{\lambda \in \lambda(A)} \left| \sum_{k=0}^{m} \gamma_k \left(\frac{\lambda}{\rho}\right)^k \right| \tag{2.7}$$

and

$$\min_{\|c\|=1} \|B_{m+1}c\| \geq (m+1)^{-1/2} \min_{\|c\|_\infty=1} \|U^*B_{m+1}c\|_\infty = (m+1)^{-1/2} \min_{\|c\|_\infty=1} \left\| \sum_{k=0}^{m} \gamma_k \sigma_k \Lambda^k e \right\|_\infty$$

$$= N^{-1/2}(m+1)^{-1/2} \min_{\|c\|_\infty=1} \max_{\lambda \in \lambda(A)} \left| \sum_{k=0}^{m} \gamma_k \left(\frac{\lambda}{\rho}\right)^k \right|. \tag{2.8}$$

It follows from (2.5), (2.7) and (2.8) that

$$\kappa(B_{m+1}) \leq N^{1/2}(m+1)^{1/2} \tilde{\kappa}_{\lambda(A)}(\{\phi_j\}_{j=0}^m). \tag{2.9}$$

Formulas (2.6) and (2.9) illustrate that the conditioning of B_{m+1} can be studied by determining the condition numbers of suitably chosen polynomial bases. Now let B_{m+1} be given by (2.1) with arbitrary parameters $\lambda_k \in \mathbb{C}$ and $r_j = U^*e$. Then inequalities analogous to (2.7) and (2.8) are valid, and we obtain, similarly with (2.9), that

$$\kappa(B_{m+1}) \leq N^{1/2}(m+1)^{1/2} \tilde{\kappa}_{\lambda(A)}(\{\phi_j\}_{j=0}^m),$$

where the polynomials $\{\phi_j\}_{j=0}^m$ are of Newton form

$$\phi_j(\zeta) = c_j \prod_{l=1}^{j} (\zeta - \lambda_l), \qquad 0 \leq j \leq m. \tag{2.10}$$

Theoretical and computational results in Reichel (1985, 1990) show that a polynomial basis of Newton form with a suitable choice of parameters λ_l can be much better conditioned than a basis of power form. In particular, in Reichel (1990) the Newton basis is found to be fairly well-conditioned on a compact set $S \subset \mathbb{C}$ when the λ_l are chosen to be Leja points (defined below) for S. This suggests that a good choice of parameters λ_l in (2.10) would be Leja points for $\lambda(A)$. However, $\lambda(A)$ is, in general, not explicitly known. We therefore choose λ_l to be Leja points for $\lambda(H_m)$, where $H_m \in \mathbb{R}^{m \times m}$ is the upper Hessenberg matrix defined by (1.8), and is computed by the Arnoldi process in Algorithm 1.1.

We note in passing that other choices of fairly well-conditioned polynomial

bases are possible. For instance, Joubert & Carey (1991) and de Sturler (1991) propose the use of a basis of Chebyshev polynomials associated with an ellipse that contains $\lambda(A)$. The conditioning of Chebyshev polynomial bases are investigated in Gautschi (1977) and Reichel (1985).

Let S be a compact set in $\mathbb{C}$. A point set $\{\zeta_j\}_{j=1}^m \subset S$ is said to be a set of *Leja points* for S if

$$|\zeta_1| = \max_{\zeta \in S} |\zeta| \tag{2.11a}$$

$$\prod_{l=1}^{j} |\zeta_{j+1} - \zeta_l| = \max_{\zeta \in S} \prod_{l=1}^{j} |\zeta - \zeta_l|, \qquad j = 1, 2, \ldots, m-1. \tag{2.11b}$$

Note that the set of Leja points for a given set S might not be unique. Asymptotic properties of Leja points for compact sets of infinite cardinality were first studied by Edrei (1939) and Leja (1957). In particular, if S is an interval, then the Leja points for S are distributed roughly like zeros of Chebyshev polynomials for S, and if instead S is a disk, then the Leja points for S are uniformly distributed on the boundary of S; see Edrei (1939), Leja (1957), Reichel (1990, 1991).

3. A new GMRES implementation

This section describes our implementation of the GMRES method. We first note that the GMRES method does not require the matrix A to be stored. When using the implementations by Saad & Schultz (1986) (Algorithm 1.1) or by Walker (1988), it suffices to compute matrix–vector products Au for certain vectors $u \in \mathbb{R}^N$. In many applications the vector $\delta(A - \tau I)u$, where δ and τ are real constants, can be evaluated almost as rapidly as the vector Au. Matrices A with this property arise from the discretization of partial differential and integral equations. This is illustrated by the matrices used in the computed examples of Section 4. It is convenient in our implementation to introduce a vector-valued function mult (δ, A, τ, u), whose value is $\delta(A - \tau I)u$. We use this function to evaluate Krylov subspace bases of Newton form.

Let the components of $x_{-1} \in \mathbb{R}^N$ be uniformly distributed in $[-1, 1]$ and determine a scaling factor $\beta_{-1} \in \mathbb{R}$, such that

$$\|b - A(\beta_{-1} x_{-1})\| = \min_{\beta \in \mathbb{R}} \|b - A(\beta x_{-1})\|. \tag{3.1}$$

We let $x_0 := \beta_{-1} x_{-1}$ and $r_0 := b - Ax_0$ be input vectors for Algorithm 1.1, and select the dimension m of the Krylov subspace $\mathbf{K}_m(A, r_0)$, in which the algorithm seeks a correction z_0 of x_0. The purpose of this choice of x_0 is to obtain a fairly good approximate solution that contains components of many eigenvectors of A. Algorithm 1.1 yields a new improved approximate solution x_1 and the $m \times m$ upper Hessenberg matrix H_m defined by (1.8). The spectrum $\lambda(H_m) = \{\lambda_j\}_{j=1}^m$ of H_m can be computed by standard numerical software; in the computed examples of Section 4 we used EISPACK (Smith *et al* (1970)) subroutine HQR.

Because A, B and x_0 contain real components only, so does the matrix H_m. The eigenvalues λ_j of H_m are therefore real or appear in complex conjugate pairs. First assume that $\lambda(H_m)$ is real. We can then use formulas (2.11a) and (2.11b) with $S := \lambda(H_m)$ to order the eigenvalues of H_m. If the spectrum $\lambda(H_m)$ is not real, then we modify the ordering scheme (2.11a) and (2.11b) so that for the ordered eigenvalues λ_j we have that $\mathrm{Im}\,(\lambda_j) > 0$ implies $\lambda_{j+1} = \bar{\lambda}_j$, where the bar denotes complex conjugation. This modification of (2.11a) and (2.11b) allows us to work with real vectors only.

ALGORITHM 3.1 (Modified Leja Ordering)
Input: set $\lambda(H_m)$ consisting of m points that are real or appear in complex conjugate pairs;

Output: ordered elements $\lambda_1, \lambda_2, \ldots, \lambda_m$ of $\lambda(H_m)$;

Determine $\lambda_1 \in \lambda(H_n)$ such that

$$|\lambda_1| = \max_{\lambda \in \lambda(H_m)} |\lambda|, \qquad \mathrm{Im}\,(\lambda_1) \geq 0;$$

for $j := 1, 2, \ldots, m-1$ do
  if $\mathrm{Im}\,(\lambda_j) > 0$ then
    $\lambda_{j+1} := \bar{\lambda}_j$
  else
    Determine $\lambda_{j+1} \in \lambda(H_m)$ such that

$$\prod_{l=1}^{j} |\lambda_{j+1} - \lambda_l| = \max_{\lambda \in \lambda(H_m)} \prod_{l=1}^{j} |\lambda - \lambda_l|, \qquad \mathrm{Im}\,(\lambda_{j+1}) \geq 0;$$

    If the maximum vanishes, then perturb real parts of non-ordered elements of $\lambda(H_m)$ slightly and maximize product over the new set $\lambda(H_m)$ so obtained.
  end if
end j; □

The above algorithm can be implemented so that only $O(m^2)$ floating-point operations are required.

We are now in a position to describe the computation of the matrix B_{m+1} given by (2.1). The following algorithm determines a matrix $\tilde{B}_{m+1} = [\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_{m+1}] \in \mathbb{R}^{N \times (m+1)}$ and a diagonal matrix $\tilde{D}_{m+1} = \mathrm{diag}\,(\tilde{\delta}_1, \tilde{\delta}_2, \ldots, \tilde{\delta}_{m+1})$, such that $B_{m+1} := \tilde{B}_{m+1}\tilde{D}_{m+1}$ has columns of unit length.

ALGORITHM 3.2 (Computation of $\tilde{B}_{m+1}$ and $\tilde{D}_{m+1}$)
Input: m eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$ of H_m ordered by Algorithm 3.1, residual $r_1 := b - Ax_1$;

Output: matrices $\tilde{B}_{m+1}$, $\tilde{D}_{m+1}$;

$\tilde{b}_1 := r_1$; $\quad \tilde{\delta}_1 := 1/\|\tilde{b}_1\|$;
for $j := 1, 2, \ldots, m$ do
  if $\mathrm{Im}\,(\lambda_j) = 0$ then
    $\tilde{b}_{j+1} := \mathrm{mult}\,(\tilde{\delta}_j, A, \lambda_j, \tilde{b}_j)$; $\quad \tilde{\delta}_{j+1} := 1/\|\tilde{b}_{j+1}\|$;

```
    else
      if Im (λ_j) > 0 then
        b̃_{j+1} := mult (δ̃_j, A, Re (λ_j), b̃_j);
        b̃_{j+2} := mult (1, A, Re (λ_j), b̃_{j+1}) + δ̃_j Im (λ_j)^2 b̃_j;
        δ̃_{j+1} := 1/‖b̃_{j+1}‖;        δ̃_{j+2} := 1/‖b̃_{j+2}‖;
      end if
    end if
  end j;
```

□

Not counting the arithmetic operations required by the function 'mult', we see that Algorithm 3.2 requires between $2(m+1)N + O(1)$ and $2(\frac{3}{4}m+1)N + O(1)$ floating-point operations. Here and below the O(1)-terms are independent of N.

We compute the QR factorization

$$\tilde{B}_{m+1} = \tilde{Q}_{m+1}\tilde{R}_{m+1} \tag{3.2}$$

where $\tilde{Q}_{m+1} \in \mathbb{R}^{N\times(m+1)}$ is orthogonal, $\tilde{Q}_{m+1}^{\mathrm{T}}\tilde{Q}_{m+1} = I$, and $\tilde{R}_{m+1} \in \mathbb{R}^{(m+1)\times(m+1)}$ is upper triangular. We can choose an algorithm for computing the decomposition (3.2) that is suitable for the computer at hand; see, e.g., Chu & George (1989a,b), Dongarra *et al* (1986, 1991), Luk (1986), Pothen & Raghavan (1989) and references therein for a variety of algorithms. The standard sequential algorithm for computing this factorization uses Householder transformations and requires $2(m+1)^2N + O(1)$ floating-point operations (see Golub & Van Loan (1989, p 212)) and this is the computationally most demanding step in each iteration. In the computed examples in Section 4 we compute the QR factorization (3.2) by using Householder transformations, which are implemented with level 1 or level 2 BLAS (Golub & Van Loan (1989)).

The next step of our implementation is to compute a QR factorization of the matrix AB_m, where B_m is given by (1.11). This can be accomplished with fairly little work, given the factorization (3.2). We use the fact that the columns $\tilde{b}_j$ of $\tilde{B}_{m+1}$ satisfy a recursion relation. If $\lambda_j \in \mathbb{R}$ then

$$\tilde{b}_{j+1} = \tilde{\delta}_j(A - \lambda_j I)\tilde{b}_j \tag{3.3a}$$

and if $\mathrm{Im}\,(\lambda_j) > 0$, then

$$\begin{aligned} \tilde{b}_{j+1} &= \tilde{\delta}_j(A - \mathrm{Re}\,(\lambda_j)I)\tilde{b}_j \\ \tilde{b}_{j+2} &= \tilde{\delta}_j(A - \lambda_j I)(A - \bar{\lambda}_j I)b_j \end{aligned} \tag{3.3b}$$

where $\tilde{\delta}_j = 1/\|\tilde{b}_j\|$. Let $\tilde{Q}_{m+1} := [\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_{m+1}]$ and $\tilde{R}_{m+1} := [\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_{m+1}] = [\tilde{\rho}_{kj}]_{k,j=1}^{m+1}$. Then by (3.2), for $1 \leq j \leq m+1$,

$$\tilde{B}_{m+1}e_j = \tilde{b}_j = \tilde{Q}_{m+1}\tilde{r}_j = \sum_{k=1}^{j} \tilde{q}_k\tilde{\rho}_{kj}. \tag{3.4}$$

First assume that $\lambda_j \in \mathbb{R}$. Then, by (3.3a) and (3.4), for $1 \leq j \leq m$,

$$\begin{aligned} AB_{m+1}e_j &= A\tilde{B}_{m+1}\tilde{D}_{m+1}e_j = \tilde{\delta}_j A\tilde{b}_j = \tilde{b}_{j+1} + \tilde{\delta}_j\lambda_j\tilde{b}_j \\ &= \sum_{k=1}^{j+1} \tilde{q}_k\tilde{\rho}_{k,j+1} + \tilde{\delta}_j\lambda_j\sum_{k=1}^{j} \tilde{q}_k\tilde{\rho}_{kj} = \sum_{k=1}^{j} \tilde{q}_k(\tilde{\rho}_{k,j+1} + \tilde{\delta}_j\lambda_j\tilde{\rho}_{kj}) + \tilde{q}_{j+1}\tilde{\rho}_{j+1,j+1}. \end{aligned} \tag{3.5}$$

Let

$$\begin{aligned} \hat{\rho}_{kj} &:= \tilde{\rho}_{k,j+1} + \tilde{\delta}_j \lambda_j \tilde{\rho}_{kj}, \qquad 1 \leq k \leq j \\ \hat{\rho}_{j+1,j} &:= \tilde{\rho}_{j+1,j+1}. \end{aligned} \tag{3.6}$$

Then (3.5) can be written

$$AB_{m+1}e_j = \sum_{k=1}^{j+1} \tilde{q}_k \hat{\rho}_{kj}. \tag{3.7}$$

If instead $\operatorname{Im}(\lambda_j) > 0$ for $1 \leq j \leq m-1$, then we obtain from (3.3b) and (3.4) that

$$\begin{aligned} AB_{m+1}e_j = A\tilde{B}_{m+1}\tilde{D}_{m+1}e_j &= \tilde{\delta}_j A\tilde{b}_j = \tilde{b}_{j+1} + \tilde{\delta}_j \operatorname{Re}(\lambda_j)\tilde{b}_j \\ &= \sum_{k=1}^{j+1} \tilde{q}_k \tilde{\rho}_{k,j+1} + \tilde{\delta}_j \operatorname{Re}(\lambda_j) \sum_{k=1}^{j} \tilde{q}_k \tilde{\rho}_{kj} \\ &= \sum_{k=1}^{j} \tilde{q}_k \left(\tilde{\rho}_{k,j+1} + \tilde{\delta}_j \operatorname{Re}(\lambda_j)\tilde{\rho}_{kj}\right) + \tilde{q}_{j+1}\tilde{\rho}_{j+1,j+1} \end{aligned} \tag{3.8}$$

and

$$\begin{aligned} AB_{m+1}e_{j+1} = A\tilde{B}_{m+1}\tilde{D}_{m+1}e_{j+1} &= \tilde{\delta}_{j+1} A\tilde{b}_{j+1} \\ &= \tilde{\delta}_{j+1}\left(\tilde{b}_{j+2} + \operatorname{Re}(\lambda_j)\tilde{b}_{j+1} - \tilde{\delta}_j \operatorname{Im}(\lambda_j)^2 \tilde{b}_j\right) \\ &= \sum_{k=1}^{j} \tilde{q}_k \tilde{\delta}_{j+1}\left(\tilde{\rho}_{k,j+2} + \operatorname{Re}(\lambda_j)\tilde{\rho}_{k,j+1} - \tilde{\delta}_j \operatorname{Im}(\lambda_j)^2 \tilde{\rho}_{kj}\right) \\ &\quad + \tilde{q}_{j+1}\tilde{\delta}_{j+1}\left(\tilde{\rho}_{j+1,j+2} + \operatorname{Re}(\lambda_j)\tilde{\rho}_{j+1,j+1}\right) + \tilde{q}_{j+2}\tilde{\delta}_{j+1}\tilde{\rho}_{j+2,j+2}. \end{aligned} \tag{3.9}$$

Let

$$\begin{aligned} \hat{\rho}_{kj} &:= \tilde{\rho}_{k,j+1} + \tilde{\delta}_j \operatorname{Re}(\lambda_j)\tilde{\rho}_{kj}, \qquad 1 \leq k \leq j \\ \hat{\rho}_{j+1,j} &:= \tilde{\rho}_{j+1,j+1} \\ \hat{\rho}_{k,j+1} &:= \tilde{\delta}_{j+1}\left(\tilde{\rho}_{k,j+2} + \operatorname{Re}(\lambda_j)\tilde{\rho}_{k,j+1} - \tilde{\delta}_j \operatorname{Im}(\lambda_j)^2 \tilde{\rho}_{kj}\right), \qquad 1 \leq k \leq j \\ \hat{\rho}_{j+1,j+1} &:= \tilde{\delta}_{j+1}\left(\tilde{\rho}_{j+1,j+2} + \operatorname{Re}(\lambda_j)\tilde{\rho}_{j+1,j+1}\right) \\ \hat{\rho}_{j+2,j+1} &:= \tilde{\delta}_{j+1}\tilde{\rho}_{j+2,j+2}. \end{aligned} \tag{3.10}$$

Then we have

$$AB_{m+1}e_j = \sum_{k=1}^{j+1} \tilde{q}_k \hat{\rho}_{kj} \qquad \text{and} \qquad AB_{m+1}e_{j+1} = \sum_{k=1}^{j+2} \tilde{q}_k \hat{\rho}_{k,j+1} \tag{3.11}$$

and it follows from (3.8) and (3.11) that the $(m+1) \times m$ matrix $\hat{R} = [\hat{\rho}_{jk}]$ satisfies

$$AB_m = \tilde{Q}_{m+1}\hat{R} \tag{3.12}$$

where $\hat{\rho}_{jk} := 0$, if $1 < j+1 < k \leq m+1$. A QR factorization $\hat{R} = \hat{Q}R$, where $\hat{Q} \in \mathbb{R}^{(m+1)\times m}$, $\hat{Q}^{\mathrm{T}}\hat{Q} = I$ and $R \in \mathbb{R}^{m \times m}$ is upper triangular, can be determined by using only m Givens rotations. This factorization and (3.12) are used to solve (1.9″). We summarize our implementation.

ALGORITHM 3.3 (A new GMRES implementation)

Choose initial vector x_{-1} as described in the beginning of this section;

Solve minimization problem (3.1) for β_{-1} and form $x_0 := \beta_{-1}x_{-1}$; $r_0 := b - Ax_0$;

Select the dimension m of the Krylov subspaces to be used;

Apply Algorithm 1.1 to determine a new approximation x_1 and an $m \times m$ Hessenberg matrix H_m;

Compute residual $r_1 := b - Ax_1$;

Compute the spectrum $\lambda(H_m) = \{\lambda_j\}_{j=1}^m$ and order the λ_j with Algorithm 3.1;

for $j := 1, 2, \ldots$ until $\|r_j\|$ is sufficiently small, do
  Compute columns of $\tilde{B}_{m+1}$ by Algorithm 3.2;
  Compute QR factorization of $\tilde{B}_{m+1} = \tilde{Q}_{m+1}\tilde{R}_{m+1}$;
  Compute the $(m+1) \times m$ Hessenberg matrix $\hat{R}$ by (3.6) and (3.10), so that $AB_m = \tilde{Q}_{m+1}\hat{R}$;
  Compute QR factorization of $\hat{R}$, solve (1.9″) for d_j;
  Compute $x_{j+1} := x_j + \tilde{B}_m\tilde{D}_m d_j$ and $r_{j+1} := b - Ax_{j+1}$;
end j. □

The number of matrix–vector products with the matrix A is the same for all GMRES implementations available. We therefore ignore the arithmetic operations required by the subroutine 'mult', which computes matrix–vector products with matrix A. This leads to a count of between $(2m^2 + 8m + 5)N + O(1)$ and $(2m^2 + 7{\cdot}5m + 5)N + O(1)$ floating-point operations for each iteration of Algorithm 3.3. The corresponding operation count for the implementation by Saad and Schultz (1986) (Algorithm 1.1) is $(2m^2 + 7m + 4)N + O(1)$. Walker's implementation of the GMRES algorithm (Walker (1988, 1989)) requires a factor 2–3 more arithmetic work than the implementation by Saad and Schultz (1986). We refer to Walker (1988) for a detailed operation count for Walker's implementation.

We remark that in Algorithm 3.3 we only reduce the given matrix A to an upper Hessenberg matrix H_m once by the Arnoldi process (in Algorithm 1.1) and compute the eigenvalues $\{\lambda_j\}_{j=1}^m$ of H_m, which after ordering are used to determine the Newton bases that define columns of the matrices $\tilde{B}_{m+1}$ generated in the j-loop. Numerous computed examples, some of which are presented in Section 4, indicate that if we choose the initial vector x_0 so that it contains components of many eigenvectors, then the λ_j so obtained yield Newton bases of the Krylov subspaces $\mathbf{K}_m(A, r_j)$, $j \geq 1$, that give rise to fairly well-conditioned matrices $\tilde{B}_{m+1}$. In all computed examples we chose x_0 to have randomly generated uniformly distributed entries, as described at the beginning of this section. With this choice of x_0 we obtained eigenvalues λ_j of H_m that gave fairly well-conditioned matrices $\tilde{B}_{m+1}$ in the j-loop. However, it cannot be ruled out that for certain initial vectors x_0 or for certain matrices A, some matrix $\tilde{B}_{m+1}$ generated in the j-loop of Algorithm 3.3 is ill-conditioned enough to affect the convergence of the iterates x_j. In this case, we have to return to Algorithm 1.1

and again reduce A to an upper Hessenberg matrix. Let L be the union of the eigenvalues of this Hessenberg matrix and the eigenvalues previously computed, and apply Algorithm 3.1 to determine an ordering of the elements of L. Let $\{\lambda_j\}_{j=1}^m$ denote the first m elements in the sequence so obtained. We now use these λ_j to define subsequent Newton bases in the j-loop. This modification of Algorithm 3.3 is easy to implement in a production code, but clutters the presentation and has therefore been omitted from Algorithm 3.3. Our numerical experience suggests that Algorithm 3.3 as presented is adequate for many problems. The computed examples of Section 4 are based on Algorithm 3.3 as stated above, i.e., in each experiment only one set of eigenvalues $\{\lambda_j\}_{j=1}^m$ is computed.

4. Numerical examples

We describe some numerical experiments with two Newton basis GMRES implementations and compare them with the GMRES implementation by Saad & Schultz (1986) (Algorithm 1.1). The purpose of the examples is to illustrate the stability of the Newton basis implementations, and to present some timings. The stability and CPU timings were performed on an IBM RISC System/6000 model 550 workstation using single-precision arithmetic.

We derived our test problem by discretizing the boundary value problem

$$\begin{aligned} -\Delta u + 2p_1 u_x + 2p_2 u_y - p_3 u &= f \quad \text{in} \quad \Omega \\ u &= 0 \quad \text{on} \quad \partial\Omega \end{aligned} \tag{4.1}$$

by finite differences, where Ω is the unit square $\{(x, y) \in \mathbb{R}^2, 0 \leq x, y \leq 1\}$ and p_1, p_2, p_3 are positive constants. The right-hand side function $f(x, y)$ was chosen so that $u(x, y) = xe^{xy} \sin(\pi x) \sin(\pi y)$ solves (4.1). More precisely, (4.1) is discretized by centred finite differences on a uniform $(n+2) \times (n+2)$ grid (including grid points on the boundary) and we use the standard five-point stencil to approximate the Laplacian. Let $h := 1/(n+1)$. After scaling by h^2, the algebraic linear system of equations of order n^2 obtained from (4.1) can be written as (1.1) with $N = n^2$, where a typical equation for the unknown $u_{ij} \approx u(ih, jh)$ is given by

$$(4-\sigma)u_{ij} - (1+\beta)u_{i-1,j} - (1-\beta)u_{i+1,j} - (1+\gamma)u_{i,j-1} - (1-\gamma)u_{i,j+1} = h^2 f_{ij}.$$

Here $\beta := p_1 h$, $\gamma := p_2 h$, $\sigma := p_3 h^2$ and $f_{ij} := f(ih, jh)$. No preconditioner was used in order to keep the issues of interest clear. Generally, however, a preconditioner should be used for the solution of linear systems of equations that arise from the discretization of partial differential equations; see Anderson & Saad (1989), Saad (1989), Saad & Schultz (1986), Walker (1988) for discussions and illustrations.

Our first three examples compare the accuracy achieved by the GMRES implementation of Saad & Schultz (1986), which uses the Arnoldi process implemented by the MGS method, with the accuracy obtained with our Newton basis implementation. We refer to the GMRES implementation by Saad & Schultz (1986) as 'Arnoldi–GMRES'. We refer to our Newton basis implementation

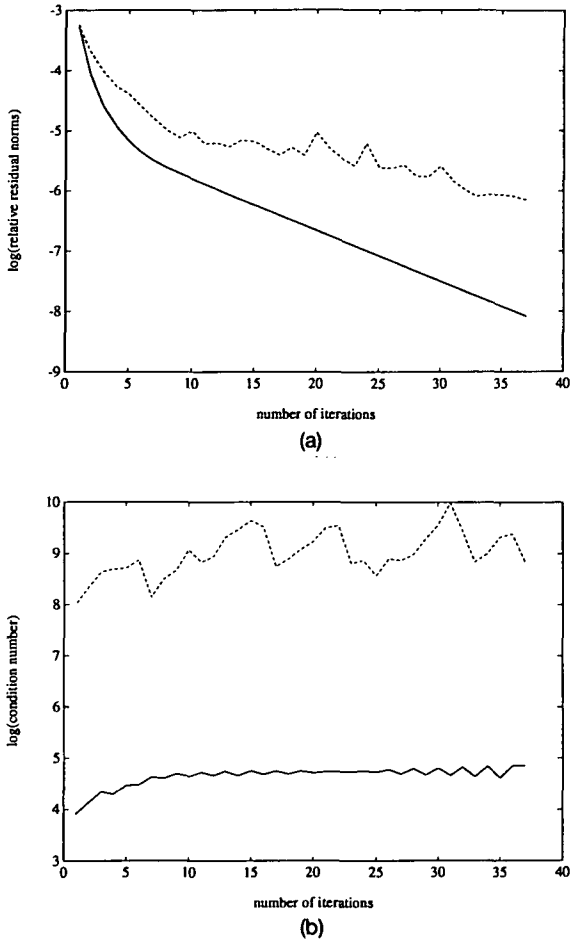

FIG. 1. (a) Convergence comparison of Arnoldi–GMRES (dotted curve), Newton–GMRES (solid curve) and power–GMRES (dashed curve). The graphs for Arnoldi–GMRES and Newton–GMRES coincide. (b) Condition numbers of the matrices B_m.

summarized by Algorithm 3.3 as 'Newton–GMRES'. If we choose all the λ_j to be equal to zero in the latter implementation, then the Newton basis becomes a scaled power basis. We refer to the GMRES implementation using this power basis as 'power–GMRES'.

EXAMPLE 4.1 In this example, we let $n = 63$, i.e., the order N of the linear system is 3969, and we choose the constants $p_1 = 1$, $p_2 = 1$, $p_3 = 20$ and $m = 20$. Fig. 1(a) shows the logarithm of the quotient of norms of residual vectors $\log_{10}(\|r_k\|/\|r_0\|)$ evaluated for $k = 1, 2, \ldots, 25$ for Newton–GMRES (solid curve), Arnoldi–GMRES (dotted curve), and power–GMRES (dashed curve). The Newton–GMRES and Arnoldi–GMRES behave roughly the same (the curves are indistinguishable), but clearly, the power–GMRES implementation is inferior. This is due to the fact that

the columns of the matrix B_m for the power basis are numerically linearly dependent. This is illustrated by Fig. 1(b), which shows $\log_{10} \kappa(B_m)$, where the matrix B_m is defined by (1.11). The solid curve displays the condition number for the Newton basis, and the dashed curve shows the condition number for the power basis. We see that $\kappa(B_m)$ is significantly smaller for the Newton basis than for the power basis. In this example all the Leja-sorted points for forming the Newton basis are real.

EXAMPLE 4.2 This example differs from Example 4.1 only in that now $m = 30$. Fig. 2(a) is analogous with Fig. 1(a), and shows $\log_{10}(\|r_k\|/\|r_0\|)$ evaluated for $k = 1, 2, \ldots, 25$ for Newton–GMRES (solid curve), Arnoldi–GMRES (dotted curve) and power–GMRES (dashed curve). The Newton–GMRES and Arnoldi–GMRES

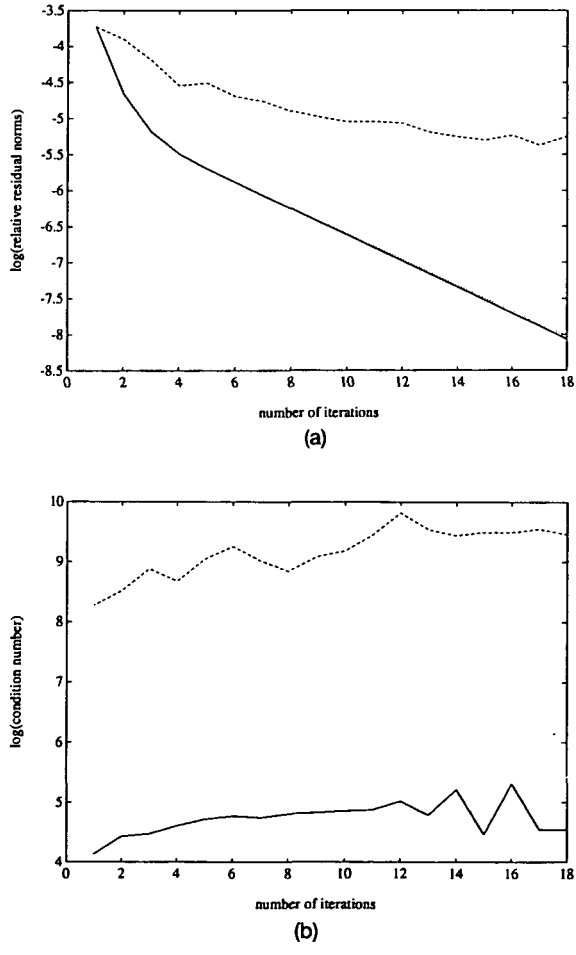

FIG. 2. (a) Convergence comparison of Arnoldi–GMRES (dotted curve), Newton–GMRES (solid curve) and power–GMRES (dashed curve). The graphs for Arnoldi–GMRES and Newton–GMRES almost coincide. (b) Condition numbers of the matrices B_m.

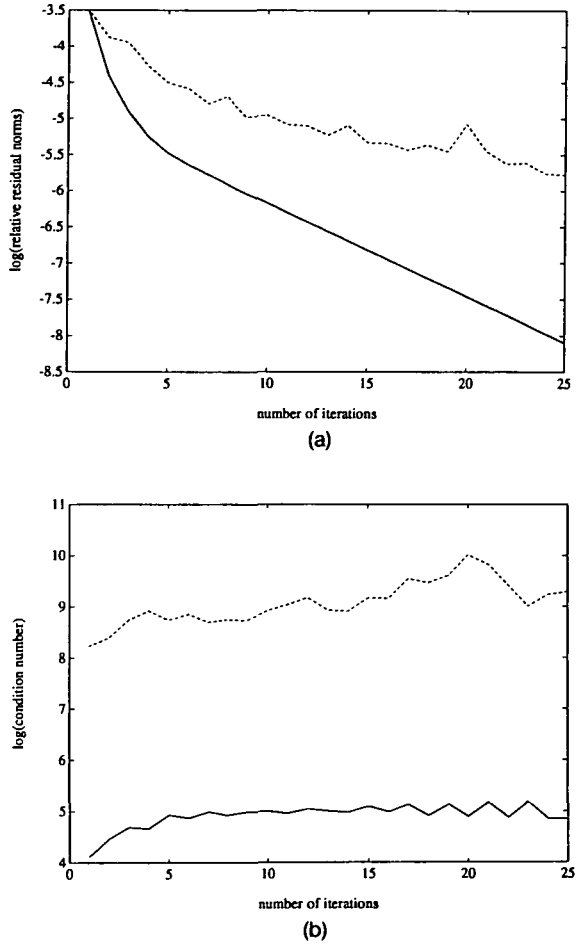

FIG. 3. (a) Convergence comparison of Arnoldi–GMRES (dotted curve), Newton–GMRES (solid curve) and power–GMRES (dashed curve) implementations. The graphs for Arnoldi–GMRES and Newton–GMRES coincide. (b) Condition numbers of the matrices B_m.

implementations yield residual vectors of nearly the same norm. Fig. 2(b) is analogous with Fig. 1(b) and shows the condition number of the matrix B_m for the Newton basis (solid curve) and for the power basis (dashed curve).

EXAMPLE 4.3 We keep $n = 63$, but choose the constants $p_1 = 2$, $p_2 = 4$, $p_3 = 30$ and $m = 25$. Fig. 3(a) is analogous with Fig. 1(a), and shows $\log_{10}(\|r_k\|/\|r_0\|)$ evaluated for $k = 1, 2, \ldots, 25$ for the Newton–GMRES implementation (solid curve), Arnoldi–GMRES implementation (dotted curve) and power–GMRES implementation (dashed curve). Fig. 3(b) shows the condition number of the matrix B_m for the Newton basis (solid curve) and power basis (dashed curve). Some of the points λ_j that define the Newton basis appear in complex conjugate pairs.

EXAMPLE 4.4 This example shows some timings comparing two Newton–GMRES

TABLE 1
CPU times in seconds on an IBM RISC System/6000 model 550 workstation

N	m	No of iterations	Newton-1	Newton-2	Arnoldi
3969	40	30	44·81	37·93	41·65
4900	40	30	55·15	46·21	51·27
6400	40	30	72·23	61·38	66·88
8100	40	30	91·63	77·50	84·80
10000	40	30	119·83	96·25	108·72

implementations with the Arnoldi–GMRES implementation. The basic linear algebra subprograms (BLAS) used for the computation of inner products of two vectors and of matrix–vector products are optimized for the workstation. Table 1 displays CPU times required by the different GMRES implementations.

The first column of the table is the order of the matrix A, the second column is the size of Krylov subspace. The number of iterations is shown in column 3. Column 4 (Newton-1) displays the total CPU time in seconds required by a Newton–GMRES implementation that uses a subroutine for QR factorization coded with level-1 BLAS. Column 5 (Newton-2) shows the total CPU time in seconds required by a Newton–GMRES implementation that uses a subroutine for QR factorization coded with level-1 and level-2 BLAS. The last column (Arnoldi) shows CPU times for the Arnoldi–GMRES implementation by Saad & Schultz (1986). The accuracy achieved with the Newton-1 and Newton-2 implementations is indistinguishable.

Table 1 illustrates that the performance of the subroutine for QR factorization of the $N \times (m+1)$ matrices B_{m+1} is crucial for the competitiveness of the Newton–GMRES implementation. We found that nearly 85% of the total CPU time required for the Newton–GMRES algorithm is spent computing QR factorizations of these matrices. Our implementation that uses both level-1 and level-2 BLAS (Newton-2) yields a 10% reduction in CPU time compared with the Arnoldi–GMRES implementation. We expect future performance improvements by using a more efficient scheme for the QR factorization of B_{m+1}.

5. Summary and extensions

This paper describes a new implementation of the popular GMRES method for solving large sparse non-symmetric linear systems of equations. In the new implementation the vector–vector operations of the MGS method used in the GMRES implementation by Saad & Schultz (1986) are replaced by the task of computing a QR factorization of a dense $N \times (m+1)$ matrix. The numerical examples show the same numerical stability and accuracy of the new implementation as of the implementation in Saad & Schultz (1986).

Our new scheme allows more flexibility in the implementation of the GMRES method, because it allows the use of a subroutine for computing a QR

factorization that is tailored for the computer at hand. This part of the computation is by far the most demanding in CPU time. The competitiveness of our scheme depends on the availability of an efficient subroutine for the computation of a QR factorization of a dense matrix of size $N \times (m+1)$, where, typically, $N \gg m$. We expect to be able to reduce the CPU time required by our sequential implementation further in the future, by using a more efficient subroutine for computing a QR factorization of dense matrices of this special form.

The possibility of choosing a QR factorization scheme would appear to be of particular value for a parallel implementation of the GMRES algorithm. Such an implementation can be based on the QR factorization schemes of Chu & George (1989a,b). In these schemes, the $N \times (m+1)$ matrix $\tilde{B}_{m+1}$ is first partitioned into p blocks

$$\tilde{B}_{m+1} = \begin{pmatrix} \tilde{B}_1 \\ \tilde{B}_2 \\ \vdots \\ \tilde{B}_p \end{pmatrix} \begin{matrix} n_1 \\ n_2 \\ \vdots \\ n_p \end{matrix}$$

and the QR factorizations of the $n_i \times (m+1)$ matrices $\tilde{B}_i$, $i = 1, 2, \ldots, p$, are computed simultaneously. Then the triangular blocks are reduced to obtain the desired QR factorization of $\tilde{B}_{m+1}$. The latter computations can also be carried out in parallel. Timings for a parallel implementation of the Newton–GMRES method on an IBM 3090–600VF computer are presented in Calvetti *et al* (1993).

Acknowledgement

The authors would like to express thanks to the referees, whose comments led to improvements in the presentation.

References

ANDERSON, E., & SAAD, Y. 1989 Preconditioned conjugate gradient methods for general sparse matrices on shared memory computers. *Parallel Processing for Scientific Computing* (G. Rodrigue, ed). Philadelphia: SIAM, pp 88–92.

CALVETTI, D., PETERSEN, J., & REICHEL, L. 1993 A parallel implementation of the GMRES method. *Numerical Linear Algebra* (L. Reichel, A. Ruttan and R. S. Varge, eds). Berlin: de Gruyter, pp 31–46.

CHRONOPOULOS, A., & KIM, S. K. 1990 *S-step* ORTHOMIN *and* GMRES *implemented on parallel computers.* Department of Computer Science, University of Minnesota, *TR*-90-15.

CHU, E., & GEORGE, A. 1989a QR factorization of a dense matrix on a shared memory multiprocessor. *Parallel Comput.* **11,** 55–71.

CHU, E., & GEORGE, A. 1989b QR factorization of a dense matrix on a hypercube multiprocessor. *SIAM J. Sci. Stat. Comput.* **11,** 990–1028.

DE STURLER, E. 1991 A parallel variant of GMRES(m). *Proc. 13th World Congress on Computation and Applied Mathematics* (*Dublin, IMACS*'91) pp 682–3.

DONGARRA, J. J., DUFF, I. S., SORENSEN, D. C., & VAN DER VORST, H. A. 1991 *Linear System Solving on Vector and Shared Memory Computers.* Philadelphia: SIAM.

DONGARRA, J. J., SAMEH, A., & SORENSEN, D. C. 1986 Implementation of some concurrent algorithms for matrix factorization. *Parallel Comput.* **3,** 25–34.
EDREI, A. 1939 Sur les déterminants récurrents et les singularités d'une fonction donneé par son développement de Taylor. *Composito Math.* **7,** 20–88.
FISCHER, B., & REICHEL, L. 1989 Newton interpolation in Fejér and Chebyshev points. *Math. Comput.* **53,** 265–78.
GAUTSCHI, W. 1977 The condition of orthogonal polynomials. *Math. Comput.* **26,** 923–4.
GAUTSCHI, W. 1979 Condition of polynomials in power form. *Math. Comput.* **33,** 343–52.
GAUTSCHI, W. 1984 Questions of numerical condition related to polynomials. *Studies in Numerical Analysis* (G. H. Golub, ed). Mathematics Association of America.
GOLUB, G. H., & VAN LOAN, C. F. 1989 *Matrix Computations* (2nd edn). Baltimore, MD: Johns Hopkins University Press.
HINDMARSH, A. C., & WALKER, H. F. 1986 Note on a Householder implementation of the GMRES method. *Report UCID-20899,* Lawrence Livermore National Laboratory.
JOUBERT, W. D., & CAREY, G. F. 1991 Parallelizable restarted iterative methods for nonsymmetric linear systems. *Report CNA-251,* Center for Numerical Analysis, The University of Texas at Austin.
LEJA, F. 1957 Sur certaines suites liées aux ensemble plans et leur application à la representation conforme. *Ann. Polon. Math.* **4,** 8–13.
LUK, F. T. 1986 A rotation method for computing the *QR* decomposition. *SIAM J. Sci. Stat. Comput.* **7,** 452–9.
POTHEN, A., & RAGHAVAN, P. 1989 Distributed orthogonal factorization: Givens and Householder algorithms. *SIAM J. Sci. Stat. Comput.* **10,** 1113–34.
REICHEL, L. 1985 On polynomial approximation in the complex plane with application to conformal mapping. *Math. Comput* **44,** 425–33.
REICHEL, L. 1990 Newton interpolation at Leja points. *BIT* **30,** 332–46.
REICHEL, L. 1991 The application of Leja points to Richardson iteration and polynomial preconditioning. *Linear Algebra Appl.* **154–156,** 389–414.
SAAD, Y. 1989 Krylov subspace methods on supercomputers. *SIAM J. Sci. Stat. Comput.* **10,** 1200–32.
SAAD, Y., & SCHULTZ, M. H. 1986 GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* **7,** 856–69.
SMITH, B. T., BOYLE, J. M., IKEBE, Y., KLEMA, V. C., & MOLER, C. B. 1970 *Matrix Eigensystem Routines: EISPACK Guide* (2nd edn). New York: Springer.
WALKER, H. F. 1988 Implementation of the GMRES method using Householder transformations. *SIAM J. Sci. Stat. Comput.* **9,** 152–63.
WALKER, H. F. 1989 Implementations of the GMRES method. *Comput. Phys. Commun.* **53,** 311–20.