Implementation of an Adaptive Algorithm for Richardson's Method

Paul E. Saylor
Department of Computer Science
1304 West Springfield Avenue
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
and
Institut für Wissenschaftliches Rechnen
Departement für Informatik
Eidgenössische Technische Hochschule
Zürich, Switzerland

and

Dennis C. Smolarski
Department of Mathematics
Santa Clara University
Santa Clara, California 95053

Dedicated to Gene Golub, Richard Varga, and David Young

Submitted by John de Pillis

ABSTRACT

We discuss the implementation of an adaptive algorithm proposed by one of us. The algorithm is a hybrid of the GMRES method and Richardson's method. Richardson's method (RM) depends on a set of parameters that are computed by minimizing the L_2 norm of a polynomial over the convex hull of eigenvalues. Execution of GMRES yields not only an approximate solution but also the approximate convex hull. RM is used to avoid storing and working with a large number of vectors as GMRES often requires. This method is also advantageous for the solution of large problems. We consider several test problems and compare our algorithm primarily with the conjugate-gradient-squared algorithm, but also with GMRES and to CG (applied to the normal equations). For many (test) problems our algorithm takes roughly 50 percent more work than the conjugate-gradient-squared algorithm, although if the matrix is either preconditioned or indefinite, our algorithm is more efficient. However, our algorithm currently imposes an undesirable burden on the user, who is invited to consider a variety of numerical parameters to manipulate, such as the number of steps of RM, in order to enhance performance: the values we suggest are only empirical.

G1. INTRODUCTION

The purpose of this paper is to describe an adaptive algorithm for Richardson's method (RM) and discuss some of its numerical properties. Effective use of RM, in our approach, depends on a sequence of iteration parameters related to the spectrum of the matrix. An *adaptive algorithm* computes successive refinements of the spectrum by starting with a crude approximation and then executing RM. If the spectrum is not well approximated, RM yields a sequence of residual errors rich in the eigenvectors whose corresponding eigenvalues are needed to improve the spectrum. This behavior of our adaptive algorithm is the same as for Manteuffel's adaptive Chebyshev algorithm. Our algorithm, which is based on the algorithm of [22, Chapter 13], began as a generalization of Manteuffel's algorithm [23]. The Manteuffel adaptive algorithm is restricted to real matrices for which the eigenvalues lie in a half plane, whereas our generalization has no such restrictions. We also use the GMRES algorithm to provide a starting vector for RM, and in this way have generalized the hybrid algorithm of [12].

Professors Golub, Varga, and Young have been leaders in the development and analysis of many fundamental numerical methods; their contributions to RM are an example, and we describe some of these briefly.

Professor Golub with a colleague [1] examined the efficiency of RM, using a parameter ordering scheme due to Lebedev and Finogenov [1], and pointed out the advantages due to its simplicity. In later work, Professor Golub studied the application to the solution of nonlinear problems [17]. One of his recent contributions to the use of RM concerns the question of accuracy in inner/outer iterations for which one of the iterations is RM [16].

Professor Varga and his colleagues in recent work [10] have treated the problem of asymptotically optimal parameters for semiiterative methods. Their work lays the theoretical basis for obtaining RM parameters that yield a smallest error norm as the number of iterations increases. In our approach, parameters are optimal if an L_2 norm of a polynomial is smallest. Other important references in the treatment of this topic are [13, 14, 28, 29, 31, 39, 42]. Also see [19, 20].

Professor Young's contributions to the theory and applications of iterative methods began at the dawn of the electronic-computer age with his profound analysis of successive overrelaxation. In the same period he showed how to accelerate RM with Chebyshev polynomials (see Young's paper in [27]), and studied the important question of ordering the acceleration parameters, revived in recent work of [32, 14, 40].

1.1. *Outline of the Paper*

In the second section an overview of the adaptive algorithm is given. A novel hybrid RM-GMRES method has been recently presented [26], with some features common to our method. We comment on both methods in Section 2.

In Section 3, the advantages of using RM are briefly discussed. As often happens in practical work, phenomena occur that lie outside the boundaries of theory. Devices and tricks, unjustifiable numerical constants, and manipulations of algorithm elements are responses of the human observer that affect performance in ways that cannot be smoothly analyzed. These are discussed in Section 4. In Section 5, the algorithm is stated that in Section 6 is applied to a set of problems, most of which arise from engineering applications. We discuss modifications of certain basic steps of the algorithm in Section 7.

1.2. *Why This Method?*

The motivation for RM is its suitability for large problems on vector and parallel processors, which we will call the *system* performance: fewer inner products, fewer vector updates, somewhat fewer arithmetic operations, and less data traffic. This especially holds for variants of RM in which one or more steps are combined [35]. Large problems, as the reader is aware, occur in the simulation of physical problems in 3D. For example, the solution of porous-media problems yields systems of one million equations [25], and even these are too small to be practical.

1.3. *Adaptive Methods*

RM is not practical unless accelerated by a sequence of iteration parameters. Our method is one of a class of adaptive methods such as the Manteuffel adaptive Chebyshev method [23], whereas the conjugate-gradient method is representative of a class of methods for which computation of the necessary parameters is an integral part of the method, Adaptive-method parameters are *external*, whereas conjugate-gradient parameters are *internal*. Examples of conjugate-gradient-like methods in the non-Hermitian case are the biconjugate-gradient method [15], the conjugate-gradient-squared method (CGS) [38], the generalized minimum-residual method (GMRES) [34], and the standard conjugate-gradient algorithm applied to the normal equations, $A^*Ax = A^*b$ (CGNR).

In our method, iteration parameters depend on the spectrum of the system matrix (which may be preconditioned). An adaptive algorithm computes iteration parameters by first computing an approximate spectrum by the power method, and only then proceeding to RM. To execute RM, an initial solution approximation is required, which could be taken to be the initial guess provided by the user. However, the computations required by the power method yield vectors from which a GMRES approximation may be computed at almost no additional cost, and this is the initial approximation we use for RM. GMRES is an important method in its own right, but in our approach it is treated as a launching pad for RM, which is meant to perform most of the work in obtaining the solution.

1.4. *Conventions and Terminology*

The importance of complex linear-algebraic systems leads us to use the more general terms *Hermitian* and *non-Hermitian* in place of symmetric and nonsymmetric. The Hermitian transpose of a matrix M will be denoted by M^*. A vector norm will always be the standard Euclidean norm, induced by the inner product $\langle u, v\rangle := \sum_{i=0}^{N} u_i \bar{v}_i$. The quantity N is the number of unknowns. An *algorithm* is a detailed description of an implementation of a method. The term *matvec* is often used and means a matrix multiplication of a vector. Many least-squares problems are mentioned; the standard abbreviation, LS, is used for least squares.

2. OVERVIEW OF THE METHOD

In this section, three computational problems are described, from the solutions of which an adaptive algorithm will be assembled in Section 4. Since the topic is Richardson's method, this exposition logically beings with a statement of RM and a definition of optimal iteration parameters, which depends on the spectrum of A. The power method is presented in the second subsection, following the description of RM. With the computation of optimal RM parameters out of the way, the algorithm is ready to execute RM, using an initial approximation computed by GMRES. In the last subsection, we compare our method with the method of Nachtigal, Reichel, and Trefethen [26].

2.1. *Richardson's Method*

Given $Ax = b$, *Richardson's method* (RM) [21] is, for $0 \leqslant i \leqslant k-1$,

$$x^{(0)} = \text{given}, \tag{1}$$

$$r^{(i)} = b - Ax^{(i)}, \tag{2}$$

$$x^{(i+1)} = x^{(i)} + \tau_i r^{(i)}, \tag{3}$$

where $\{\tau_i : i = 0, \dots, k-1\}$ is a *cycle* of *iteration parameters of period* k.

It may be shown in a straightforward way that

$$r^{(i)} = R_i(A) r^{(0)},$$

where

$$R_i(\lambda) = \prod_{j=1}^{i-1} (1 - \tau_j \lambda).$$

A polynomial such as R_i for which

$$R_i(0) = 1 \tag{4}$$

is called a *residual polynomial*. Any residual polynomial yields a set of τ-parameters; optimality of these parameters is taken up next.

2.2. *Optimal Residual Polynomials*

From

$$r^{(k)} = R_k(A) r^{(0)}$$

it follows that

$$\|r^{(k)}\| \leqslant \| R_k(A) \| \|r^{(0)}\| \leqslant \|S^{-1}\| \| R_k(\Lambda) \| \|S\| \|r^{(0)}\| \tag{5}$$

if A is diagonalizable, and

$$A = S^{-1} \Lambda S$$

is the diagonalization, where $\Lambda = \mathrm{diag}(\lambda_i)$ is the diagonal matrix of eigenvalues.

DEFINITION 1. An (L_2-)*optimal* residual polynomial is one for which $R_k(\Lambda)$ is minimum in the sense that

$$\frac{1}{L} \int_\Gamma | R_k(\lambda) |^2 w(\lambda) |d\lambda| = \text{minimum}, \tag{6}$$

where Γ is a contour enclosing the spectrum of A, $L = \int_\Gamma |d\lambda|$ is a normalization factor, and w is a positive weight function.

There is of course no unique optimal polynomial, since optimality depends not only on the weight function but also on the contour.

In practice, the measure in (6) is discrete, and the integral reduces to a sum

$$\|R_k\|_w^2 = \frac{1}{M_w} \sum_{i=1}^{M} |R_k(\zeta_i)|^2 w(\zeta_i) m(\zeta_i), \tag{7}$$

where

$$M_w = \sum_{i=1}^{M} w(\zeta_i) m(\zeta_i),$$

ζ_i is a point on Γ, and m is a measure. [Note that neither (6) nor (7) is an induced norm for matrix $R_k(\Lambda)$.]

The condition (4) means that

$$R_k(\lambda) = 1 + \sum_{i=0}^{n} \rho_i \lambda^i.$$

Equation (6) reduces to a least-squares problem in the coefficients ρ_i,

$$M_w \|R_k\|_w^2 = \| F(\bar{\rho}_1, \dots, \bar{\rho}_k)^* - y \|^2 = \text{minimum}, \tag{8}$$

where $F = (f_{ij})$ is the $M \times k$ matrix defined by

$$f_{ij} = \zeta_i^j [w(\zeta_i) m(\zeta_i)]^{1/2},$$

and

$$y = \left(-[w(\zeta_1) m(\zeta_1)]^{1/2}, \dots, -[w(\zeta_M) m(\zeta_M)]^{1/2} \right)^* .$$

Another method and algorithm to determine the optimal residual polynomial are described in [36]. In our experiments, the method based on solving (8) was equally accurate.

2.3. *The Convex Hull of the Spectrum of A*

The contour Γ will be taken to be the boundary of the convex hull of the spectrum of A. Computing an approximation of the convex hull is carried out by the power method applied to A.

The method for computing eigenvalues is only outlined here. For more details see either [23] or [22, Chapter 13]. It begins with the Krylov subspace, defined next. Let s_0 be a *seed vector*.

DEFINITION 2. The *Krylov subspace* generated by s_0 is

$$V_{m+1}(A, s_0) := \operatorname{span}\{s_0, \dots, A^m s_0\}. \tag{9}$$

(Usually, the dependence on A and s_0 will not need to be shown, and we simply write V_{m+1}.) The vector s_0 will always be taken to be either the last residual obtained from the most recent execution of RM or the initial residual if RM has not yet executed.

Let $s_i = A^i s_0$. A set of approximate coefficients, $\{c_i\}$, is determined by solving the LS problem,

$$\|c_0 s_{m-j} + \cdots + c_{j-1} s_{m-1} + s_m\| = \text{minimum}, \tag{10}$$

which will be called the EGVL *LS problem*.

Computing the roots of

$$p_j(\lambda) := c_0 + \cdots + c_{j-1}\lambda^{j-1} + \lambda^j = 0 \tag{11}$$

yields approximate eigenvalues $\lambda_1, \dots, \lambda_j$ of the matrix A.

REMARK 1. The *Arnoldi method* is a method for computing eigenvalue estimates by computing a Gram-Schmidt orthogonalization for V_{j+1}, then computing the eigenvalues of a Hessenberg matrix the elements of which result from the orthogonalization. The eigenvalues of this matrix are the roots of a polynomial with the same roots as p_j. See [34] for details.

2.4. *The Generalized Minimum-Residual Method*

The eigenvalue approximations yield a contour Γ defined by the convex hull. RM could now begin with the last approximation either from the initial guess provided by the user (if RM has not yet executed) or from the most recent execution of RM, which in either case will be denoted by $x^{(0)}$; but a better choice is possible by using a linear combination of the *Krylov vectors*, chosen so as to minimize the residual. Obtaining an approximate solution by minimizing the residual over the Krylov subspace generated by $s_0 := b - Ax^{(0)}$ is equivalent to the *generalized minimum-residual method* (GMRES) [34].

The approximation $x^{(0)}$ together with the first m vectors of the Krylov subspace gives

$$x^{(m)} = x^{(0)} + \sum_{i=0}^{m-1} \alpha_i s_i,$$

where the coefficients α_i are determined by the condition that

$$\| A(x - x^{(m)}) \| = \text{minimum}. \tag{12}$$

Since

$$\begin{aligned} A(x - x^{(m)}) &= A\left[x - \left(x^{(0)} + \alpha_0 s_0 + \cdots + \alpha_{m-1} s_{m-1}\right)\right] \\ &= s_0 - (\alpha_0 s_1 + \cdots + \alpha_{m-1} s_m), \end{aligned}$$

(12) becomes

$$\| s_0 - (\alpha_0 s_1 + \cdots + \alpha_{m-1} s_m) \| = \text{minimum}, \tag{13}$$

which will be called the GMRES *LS problem.*

2.5. *The Nachtigal-Reichel-Trefethen Hybrid Method*

In a recent paper, Nachtigal, Reichel, and Trefethen [26] present a hybrid method, which, for convenience, we call the *NRT hybrid method*. It is similar to our adaptive method,[1] but with a crucial difference that we shall discuss. Nachtigal, Reichel, and Trefethen visualize their method and the general class of adaptive methods in the following striking ways (modified somewhat from [26, pp. 6, 10]. First, adaptive methods appear as

$$\begin{aligned} &\text{adaptive method: power method} \to \text{eig. est.} \to R_k \to \text{GMRES} \\ &\qquad\qquad\qquad \to R_k^{(G)} \to x^{(0)} \to \text{RM}(\text{roots of } R_k), \end{aligned}$$

in which $R_k^{(G)}$ is the GMRES residual polynomial with coefficients defined by

[1]We have chosen to call our method an adaptive method to emphasize our intellectual debt to Manteuffel, but our method as well is a hybrid of GMRES and RM.

(13), and R_k is the L_2-optimal residual polynomial with coefficients defined by (8). In the NRT hybrid method most intermediate steps are omitted:

$$\text{NRT hybrid method: GMRES} \to R_k^{(G)} \to x^{(0)} \to \text{RM}\big(\text{roots of } R_k^{(G)}\big).$$

A crucial difference between the two methods lies in the polynomials that yield RM iteration parameters. To state this more precisely, our method uses the (reciprocals of the) roots of the EGVL *polynomial*

$$p_k(\lambda) := c_0 + \cdots + c_{k-1}\lambda^{k-1} + \lambda^k,$$

whereas the NRT hybrid method uses the roots of the GMRES residual polynomial,

$$R_k^{(G)}(\lambda) = 1 - \alpha_0\lambda + \cdots - \alpha_{k-1}\lambda^k.$$

(We have made a simplification for effect. Technically, we use the roots of p_k to define a convex hull over which an L_2-optimal residual polynomial is computed. If the quantities ζ_i in the sum in (7) were roots of p_k, which arises if $M = k$, then the RM iteration parameters in our method would be the reciprocals of the roots of p_k.)

The coefficients are defined by the LS problems (11, 13). For easy comparison, these are stated here with special labels,

$$\|p_k(A)s_0 = c_0 s_0 + \cdots + c_{k-1}A^{k-1}s_0 + A^k s_0\| = \text{minimum}, \quad (\text{EGVL LS})$$

$$\|R_k^{(G)}(A)s_0 = s_0 - \alpha_0 A s_0 - \cdots - \alpha_{k-1}A^k s_0\| = \text{minimum}, \quad (\text{GMRES LS})$$

where $s_0 = r^{(0)}$.

These are complementary LS problems, and so it is not surprising that they are related. Before getting to this relation, we need to recall some terminology and make a definition. The matrix Q^*MQ is an *orthogonal section* of M if the columns of Q are orthonormal. The *field of values* of a matrix M is the set $\text{FOV}(M) := \{\langle Mv, v\rangle : \|v\| = 1\}$. Finally, we note that it is useful to make clear which space is spanned by the columns of Q and so make the following definition:

DEFINITION 3. An *orthogonal section of the matrix M over V* is any orthogonal section Q^*MQ of M for which $V = \text{span}\{q_1, \ldots, q_k\}$, where q_i is the ith column of Q.

Now we can turn to a discussion of the relation between the two LS problems. We begin with a theorem summarizing certain results and observations of Manteuffel [23].

THEOREM 1. *The* EGVL *vectors, $\{p_i(A)s_0\}_{i=0}^k$, form an orthogonal basis for V_{k+1}. The roots of $p_k(\lambda)$, which are the power-method estimates of the eigenvalues of A, are the eigenvalues of an orthogonal section of A over V_k, and as such lie in the field of values of A.*

Proof. The statement of the theorem and the proof follow [23, p. 10]. For the Krylov subspaces V_i defined in (9) we have that

$$V_1 \subset V_2 \subset \cdots \subset V_{k+1}.$$

For $1 \leqslant i \leqslant k$, assume that $A^i s_0$ is not in V_i (since otherwise the solution of the linear system is in V_i). Therefore, minimization (with respect to the Euclidean norm) of the vector

$$p_i(A)s_0 = c_0 s_0 + \cdots + c_{i-1}A^{i-1}s_0 + A^i s_0$$

is equivalent to orthogonality of $p_i(A)s_0$ to V_i. Observe that p_i depends, of course, on both A and s_0. From this it follows that the EGVL vectors $\{p_i(A)s_0\}_{i=0}^k$ are an orthogonal basis for V_{k+1}, and the roots of $p_k(\lambda) = 0$ are the eigenvalues of an orthogonal section of A over V_k. The eigenvalues of an orthogonal section of A lie in the field of values of A. ■

Now consider the GMRES LS case. Note that, due to the variable coefficient α_{k-1} of $A^k s_0$ in (GMRES LS), the GMRES vector $\{R_i^{(G)}(A)s_0\}$ is not orthogonal to V_i, and the roots of $R_i^{(G)}$ are not, at least on the basis of this reasoning, the eigenvalues of an orthogonal section of A. Nevertheless, there is a relation between the roots of $R_k^{(G)}$ and the eigenvalues of A.

THEOREM 2.

1. (*From Theorem* 1.) *The* EGVL *vectors $\{p_i(A)s_0\}_{i=0}^k$ form an orthogonal basis of V_{k+1}. The roots of $p_k(\lambda) = 0$ are the eigenvalues of an orthogonal section of A over V_k.*
2. *The polynomials $\{R_i^{(G)}(A)s_0\}_{i=0}^k$ do not, in general, form an orthogonal basis of V_{k+1}. However, there exists a set of* EGVL *vectors, to which $R_k^{(G)}(A)s_0$ belongs, that (necessarily) form an orthogonal set of basis vectors of V_{k+1}.*

3. *The roots of*[2]

$$\mu^k R_k^{(G)}(\mu^{-1}) = \mu^k - \alpha_0\mu^{k-1} - \cdots - \alpha_{k-1} = 0$$

are the eigenvalues of an orthogonal section of A^{-1} *over* $AV_k :=$ span$\{As_0, \ldots, A^k s_0\}$.

4. *The roots of* $R_k^{(G)}(\lambda) = 0$ *are reciprocals of the eigenvalues of an orthogonal section of* A^{-1} *over* AV_k.

5. *The roots of* $R_k^{(G)}(\lambda) = 0$ *are in the set* $1/\text{FOV}(A^{-1}) := \{\mu : 1/\mu \in \text{FOV}(A^{-1})\}$. *(If* $0 \in \text{FOV}(A^{-1})$, *replace* $\text{FOV}(A^{-1})$ *by* $\text{FOV}(A^{-1}) \setminus \{0\}$.)

Proof. 1: For clarity and emphasis, we have merely extracted a statement from Theorem 1.

2: The first statement has been proved in comments preceding the statement of the theorem. Now consider the EGVL vectors for the Krylov subspace $V_{k+1}(A^{-1}, A^k s_0)$, which we will denote by

$$p_{i,A^{-1}}(A^{-1})A^k s_0.$$

These form an orthogonal basis for

$$V_{k+1}(A, s_0) = V_{k+1}(A^{-1}, A^k s_0).$$

Moreover,

$$R_k^{(G)}(A)s_0 = p_{k,A^{-1}}(A^{-1})A^k s_0,$$

since each is the solution of the same LS problem.

3: Next, observe that

$$\mu^k R_k^{(G)}(\mu^{-1}) = \mu^k - \alpha_0\mu^{k-1} - \cdots - \alpha_{k-1} \equiv p_{k,A^{-1}}(\mu).$$

Since this is the EGVL polynomial for A^{-1} over the subspace $V_{k+1}(A^{-1}, A^k s_0)$, the roots of this polynomial are the eigenvalues of an orthogonal section of A^{-1} over $V_k(A^{-1}, A^k s_0) = AV_k(A, s_0)$. This proves 3.

4: The roots of $\mu^k R_k^{(G)}(\mu^{-1}) = 0$ are reciprocals of the roots of $R_k^{(G)}(\lambda) = 0$.

5: The eigenvalues of an orthogonal section of A^{-1} lie in $\text{FOV}(A^{-1})$. From 3, the roots of $R_k^{(G)}(\lambda) = 0$ therefore lie in $1/\text{FOV}(A^{-1})$. ■

[2]The expression on the left is defined only for $\mu \neq 0$, which is assumed.

REMARK 2. The crucial difference stated in Theorem 2 between our adaptive method *and the* NRT *hybrid method is that the roots we use lie in* FOV(A), and in the NRT case the roots lie in $1/\text{FOV}(A^{-1})$. This statement about our algorithm also applies to the adaptive method of Manteuffel [23] and the hybrid method of [13].

REMARK 3. The spectrum of A is contained $1/\text{FOV}(A^{-1})$. Although the field of values of a matrix is a convex set, $1/\text{FOV}(A^{-1})$ is not, in general, convex. Therefore, in general, for normal as well as nonnormal matrices, $\text{FOV}(A) \neq 1/\text{FOV}(A^{-1})$.

REMARK 4. If A is hermitian definite, then $\text{FOV}(A) = 1/\text{FOV}(A^{-1})$. The converse is not true.

3. VARIANTS OF RICHARDSON'S METHOD

In this paper, we focus on the implementation of an algorithm and, in the numerical experiments, the *arithmetic efficiency*, as measured roughly by the number of floating-point operations. However, we have been motivated by the need to solve large problems on vector and parallel processors and assert that, for this, RM is particularly advantageous. The fundamental reason is that with *a priori* computed parameters, there is no sequence of operations to synchronize, allowing the method to be rewritten in a way such that successive matvecs execute without interruption and so smoothing the course of the computation. We shall explain this further, using the grand-leap variant of RM.

The *grand-leap variant* of RM (also called a k-step method [8]) is

$$x^{(k)} = x^{(0)} + C_{k-1}(A)r^{(0)}, \tag{14}$$

where $C_{k-1}(\lambda)$ may be shown to be

$$C_{k-1}(\lambda) = \frac{1 - R_k(\lambda)}{\lambda}. \tag{15}$$

The polynomial in (15) will be called the *grand-leap polynomial*. It also arises in polynomial preconditioning as a result of the property that

$$C_{k-1}(A) \approx A^{-1}, \tag{16}$$

which may be easily derived from (15). The advantages of polynomial preconditioning have been discussed in the literature [2, 4, 33].

The advantage of the grand-leap variant is also an advantage for the variants of the k-step iterative methods presented by Chronopoulos and Gear in [7–9].

Rather than the grand-leap variant, we implemented the *leapfrog* variant [35] in our algorithm. This variant is a formulation of (2), (3) in which only $x^{(2j)}$ is computed. The grand-leap variant is easier to discuss, but the leapfrog variant makes the algorithm easier to follow.

3.1. The Advantage: Successive Matvecs

The central observation is that $C_{k-1}(A)r^{(0)}$ can be easily evaluated. To see this, assume that C_{k-1} has been factored,

$$C_{k-1}(\lambda) = g_{k-1}\prod_{j=1}^{k-1}(\lambda - \sigma_j).$$

(This assumption will be discussed below.) Therefore,

$$C_{k-1}(A)r^{(0)} = g_{k-1}\prod_{j=1}^{k-1}(A - \sigma_j I)r^{(0)}, \tag{17}$$

and RM reduces to a succession of matvecs.

3.1.1. Block Tridiagonal System Matrix.

To understand the advantage of the successive matvecs in (17), suppose A is block tridiagonal,

$$A = \begin{bmatrix} S_1 & T_1 & 0 & \cdot & \cdot & \cdot \\ T_2 & S_2 & T_3 & \cdot & & \\ 0 & T_4 & S_3 & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot & \cdot \\ \cdot & & & \cdot & \cdot & \cdot \end{bmatrix},$$

where S_i and T_i are block submatrices. To simplify the discussion, assume that the shifts, i.e., the σ_j's, are zero. We shall consider how the operations required to form $A^2 v$ may be organized. If $v = (v_1, v_2, \ldots)^T$, where v_i is a block vector, then

$$Av = \begin{bmatrix} S_1 & T_1 & 0 & \cdot & \cdot & \cdot \\ T_2 & S_2 & T_3 & \cdot & & \\ 0 & T_4 & S_3 & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot & \cdot \\ \cdot & & & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \end{bmatrix}$$

$$= \begin{bmatrix} S_1 v_1 + T_1 v_2 \\ T_2 v_1 + S_2 v_2 + T_3 v_3 \\ T_4 v_2 + S_3 v_3 + T_5 v_4 \\ \vdots \end{bmatrix}. \tag{18}$$

The first component of the next product, $A(Av)$, is

$$S_1(S_1 v_1 + T_1 v_2) + T_1(T_2 v_1 + S_2 v_2 + T_3 v_3).$$

It is easy to see the advantage in continuing with the computation of this component when the block submatrices S_1 and T_1 are in efficient memory. As the number of products increases from $A^2 v$ to $A^k v$, efficiency increases. Moreover, the work can be subdivided among k processors on a parallel computer, which is explained clearly in [33]. (Gene Golub privately pointed out these advantages to us.)

We introduce a convenient term, using the shift factors which we have ignored in discussing (18) until now.

DEFINITION 4. By *matrix pipelining* we mean the recursive computation of $(A - \sigma_k)\prod_{i=1}^{k-1}(A - \sigma_i)v$ while block elements for computing $\prod_{i=1}^{k-1}(A - \sigma_i)v$ are efficiently accessible.

3.1.2. Comments on Matrix Pipelining. If the shift factors are included, they increase the megaflop rate on Cray supercomputers when A is block tridiagonal.

On single vector processors of either a Cray-XMP or a Cray 2, the complicated programming of matrix pipelining is not necessary in order to achieve an advantage. Two or more subroutine calls of the form

1. Compute $v_2 = (A - \sigma_1)v_1$.
2. Compute $v_3 = (A - \sigma_2)v_2$.

perform more efficiently than the same number of successive steps of RM, i.e., (2), (3).[3]

3.2. *Remarks on the Representation of Polynomials*

We shall touch on some practical issues in matrix pipelining. The system performance of RM does not depend on the origin of the residual polynomial, but only on the exploitation of matrix pipelining, independently of the theoretical origins of either the residual polynomial or equivalently the grand-leap polynomial. The residual polynomial could be an L_2-optimal polynomial as we assume in our adaptive method, a GMRES residual polynomial as in the NRT hybrid method, or one resulting from conformal mapping techniques [14, 39]. However, matrix pipelining does depend on the *representation* of the polynomial.

3.2.1. Factored Form and Power Form. The grand-leap polynomial $C_{k-1}(\lambda)$ does not necessarily present itself in factored form. For example, the power-form representation i.e.,

$$C_{k-1}(\lambda) = \theta_{k-1}\lambda^{k-1} + \cdots + \theta_0,$$

may result from the same representation of the residual polynomial. To evaluate $C_{k-1}(A)r^{(0)}$ with the advantage of matrix pipelining, we turn to Horner's rule, i.e., a sequence of operations of the form

$$w_{i+1} = \theta_{k-1-i}v + Aw_i. \tag{19}$$

If the factored form is used, matrix pipelining is a sequence of operations of the form

$$v_{i+1} = (A - \sigma_i I)v_i. \tag{20}$$

Equation (19) lies midway between the simplicity of (20) and the daunting complications of the complete unfolding of the single-step form of RM in (2), (3), which yields a sequence of operations of the form

$$x_{i+1} = x_i + \tau_i(b - Ax_i). \tag{21}$$

Of course, the factored form could be computed from the power form by a rootfinder (for example, computing the eigenvalues of the companion

[3]We are indebted to M. Holst, Department of Computer Science, the University of Illinois, Urbana, for information given here on Cray vector-processor performance.

matrix). The accuracy of this unstable computation is not so critical, since the purpose of finding the roots is to evaluate the polynomial.

There is an important case in which Horner's rule is preferred, and that is to avoid mixed real and complex arithmetic, which arises when the matrix A is real. In general the roots σ_i of the grand-leap polynomial are complex, but in the case of real A, b and $x^{(0)}$ the solution and the coefficients θ_i of C_{k-1} are real. Thus (20) requires mixed arithmetic, and (19) only real arithmetic.

3.2.2. *Newton Form.* A well-known approach to RM is to derive the residual polynomial (equivalently, C_{k-1}), from conformal-mapping techniques applied to a simply connected compact region, usually thought of as containing the eigenvalues of the system matrix (or the eigenvalues of the preconditioned system matrix as the case may be). Reichel has presented an algorithm (Algorithm 4.1 in [32]) for C_{k-1} in which it is assumed to be represented as a Newton interpolating polynomial at a Leja-ordered set of points $\{\zeta_j\}_{j=1}^{k}$, i.e., an expression of the form

$$C_{k-1}(\zeta) = \sum_{j=0}^{k-1} \mu_j m_j(\zeta), \tag{22}$$

where $m_0 := \mu_0$ and

$$m_j(\zeta) := \prod_{i=0}^{j} (\zeta - \zeta_i). \tag{23}$$

Nested evaluation analogous to the way in which Horner's rule was applied to the power form yields $C_{k-1}(A)r^{(0)}$ from a sequence of evaluations of the form

$$w_{i+1} = \mu_i v + (A - \zeta_i) w_i.$$

If the factored form is preferred, it can be computed directly from the Newton form, and we outline this. From

$$z m_i(z) = z_{i+1} m_i(z) + m_{i+1}(z) \tag{24}$$

together with

$$z m_{k-2}(z) = -\frac{\mu_0}{\mu_{k-1}} m_0(z) - \cdots - \frac{\mu_{k-3}}{\mu_{k-1}} m_{k-2}(z) + z_k m_{k-2}(z) + \frac{1}{\mu_{k-1}} C_{k-1}(z)$$

we see that

$$z\begin{bmatrix} m_0(z) \\ \vdots \\ m_{k-2}(z) \end{bmatrix} = N\begin{bmatrix} m_0(z) \\ \vdots \\ m_{k-2}(z) \end{bmatrix} + \frac{1}{\mu_{k-+1}}\begin{bmatrix} 0 \\ \vdots \\ C_{k-1}(z) \end{bmatrix},$$

where

$$N = \begin{bmatrix} z_1 & 1 & & & & & & \\ 0 & z_2 & \cdot & & & & & \\ \cdot & \cdot & \cdot & \cdot & & & & \\ \cdot & & \cdot & \cdot & \cdot & & & \\ \cdot & & & \cdot & \cdot & \cdot & & \\ \cdot & & & & \cdot & \cdot & \cdot & \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 & z_{k-2} & 1 \\ -\dfrac{\mu_0}{\mu_{k-1}} & -\dfrac{\mu_1}{\mu_{k-1}} & \cdot & \cdot & \cdot & -\dfrac{\mu_2}{\mu_{k-4}} & -\dfrac{z_{k-2}-\mu_{k-2}}{\mu_{k-1}} & z_{k-1} \end{bmatrix}. \tag{25}$$

Therefore, the roots of $C_{k-1}(z)$ are the eigenvalues of N.

4. PRACTICAL ALGORITHM ISSUES

The discussion in Section 2 yields a protoalgorithm, which is a simplification of the algorithm from [22, Chapter 13]:

1. Given $x^{(0)}$, compute a Krylov subspace V_{m+1}, using $s_0 := b - Ax^{(0)}$.
2. Compute m eigenvalues. From these and any preexisting convex hull, compute a new approximate convex hull and a set of k RM parameters.,
3. Compute a GMRES approximation $x^{(m)}$.
4. Execute RM starting with $x^{(0)} := x^{(m)}$ and ending with $x^{(k)}$. Set $x^{(0)} := x^{(k)}$.
5. If not converging satisfactorily, recompute the Krylov subspace V_{m+1} with $s_0 := b - Ax^{(0)}$ and go to step 2.
6. If converged, halt.
7. If converging satisfactorily, go to step 4.

Some terminology is convenient. By *pass*, we mean that steps 4 and 5 have been executed and control has returned either to step 2 or to step 4.

We shall examine these steps from the experimental point of view and use the results to develop and justify an algorithm. A more complete discussion of experiments will be given in Section 6.

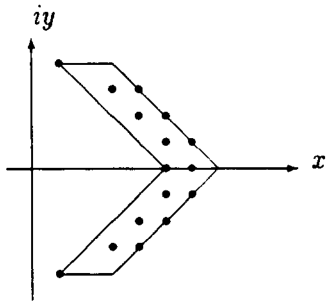

FIG. 1. Eigenvalues of the 16×16 test matrix with enclosing template for the spectrum. Eigenvalues are $1 \pm 4i$, 5, and 6. The remaining 12 eigenvalues are equally spaced along four diagonal from $2 \pm 4i$ to 6 and $3 \pm 4i$ to 7.

4.1. *The Convex Hull*

Each pass of the algorithm yields a set of eigenvalue estimates. The generation of a convex hull of these estimates may be illustrated with a simple test matrix A, a 16×16 tridiagonal matrix for which the eigenvalues are shown in Figure 1. The template placed around the eigenvalues in the figure shows the "boomerang" shape of the spectrum, characteristic of certain preconditioned elliptic difference matrices. (This example of a spectrum was suggested by H. van der Vorst [41]. Also see [37].)

In the first execution of the power method, step 2, the convex hull is shown as the smaller triangle in Figure 2. (For these experiments the number of computed eigenvalues is either $m = 3$ or $m = 2$. This choice will be discussed later.)

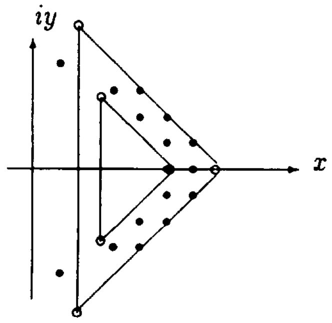

FIG. 2. Eigenvalues of the 16×16 test matrix with first computed convex hull from the algorithm. Also, the expanded hull (expansion factor of 2.0) is shown.

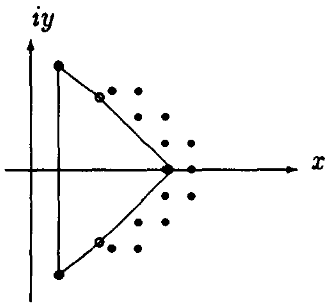

FIG. 3. Eigenvalues of the 16×16 test matrix with the second computed convex hull from the algorithm.

Figures 3 and 4 show that, in succeeding passes, outlying eigenvalues are computed from the power method and added to the previously computed convex hull. At this stage further passes did not resolve the rightmost eigenvalues accurately enough for satisfactory convergence of RM. There are two plausible explanations. One is that three steps of the power method are not sufficient to compute accurate estimates. To improve accuracy more Krylov vectors A^3s_0, $A^4s_0, \ldots$ could be computed, and only the last three used for computing a conjugate pair. The second is that the residual polynomial in RM is too large in magnitude at the eight edge of the spectrum. Techniques could be suggested for treating this phenomenon also. However, rather than either modify the Krylov subspace or modify the residual polynomial, we have taken a different approach, namely, to enlarge the convex hull uniformly by stretching each vertex of the hull by the same factor

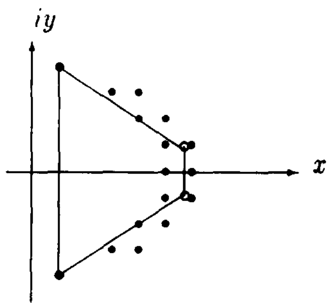

FIG. 4. Eigenvalues of the 16×16 test matrix with the third computed convex hull from the algorithm.

along a ray from the centroid. An enlarged hull is shown on Figure 2. If the polynomial is optimized over a larger region, the error corresponding to eigenvalues along the edges is damped more effectively.

We shall discuss examples in Sections 6.2 and 6.3 for which shrinking the convex hull improves performance.

4.2. *Criteria for Computing New Eigenvalue Estimates*

In the protoalgorithm, eigenvalues are updated only if the convergence of RM is judged to be unsatisfactory. One way to do this is to use

$$\frac{\|r^{(k)}\|}{\|r^{(0)}\|} \leqslant \|R_k(A)\| \tag{26}$$

with the rough estimate $\|R_k(A)\| \approx \|R_k\|_w^2$, but in practice this has not been satisfactory. Rather than perfect this criterion, we have found that after (two out of three) RM cycles automatic execution of the GMRES step has resulted in an algorithm that has performed well.

4.3. *Polynomial Preconditioning*

Until now in this paper, the adaptive algorithm has been described as RM with only enough extra computation needed to compute eigenvalues, and this effort also yielded a GMRES approximation to provide an initial approximation for RM. In practice, as stated above, we found it an advantage to use GMRES on most passes of the algorithm. This changes the nature of the algorithm. What has been described as RM may now more properly be described as a polynomial preconditioning of GMRES. For, (14) and (16) show that execution of RM is equivalent to transforming the original system $Ax = b$ into the system $C_{k-1}(A)Ax = C_{k-1}(A)b$ when the GMRES step is applied.[4] This is not strictly correct, of course, since the polynomial C_{k-1} changes from one cycle to the next, which means that the preconditioner is changing each time GMRES is applied. Nevertheless, this holds approximately and partially explains why in our numerical experiments we needed only two steps of GMRES. The polynomial from RM yields a system that is more favorable, we conjecture, either by generating a system matrix $C_{k-1}(A)A$ the Hermitian part of which is definite, or, if A is indefinite, by yielding a system matrix $C_{k-1}(A)A$ that is definite.

[4]It would be more correct to call this a *partial* polynomial preconditioning, since in the case of polynomial preconditioning the GMRES Krylov subspace would be $V_{m+1}(C_{k-1}(A)A, C_{k-1}(A)r^{(0)})$, whereas we use the GMRES Krylov subspace $V_{m+1}(A, r^{(0)})$.

4.4. *Remarks on Indefinite Matrices*

We shall say that A is *indefinite* if the convex hull of the spectrum of A includes the origin. It is well known that indefinite problems are difficult for GMRES, requiring that the approximate solution be generated from a Krylov subspace of large dimension. Indefinite problems are also difficult for RM, for the following reason: RM requires for convergence (for every matrix of a family of matrices with the same convex hull) that the residual polynomial be less than 1 in magnitude at each eigenvalue. It follows, however, from the maximum principle applied to $R_k(\lambda)$ that $|R_k(\lambda)| \geqslant 1$ on the boundary, since $R_k(0) = 1$. In order to converge when used alone and not with GMRES, it would be necessary to minimize the residual polynomial over a simply connected region not containing the origin, which implies that the difficult problem of isolating the origin by computing interior eigenvalues would have to be solved. A proposal for isolating the origin in the case when the eigenvalues are real and the matrix Hermitian is contained in [2], but it does not generalize to the complex case.

4.5. *Period of Richardson's Method*

We have observed in the numerical experiments reported in Section 6 that a period of $k = 8$ was effective over a range of test problems.

5. THE ALGORITHM

ALGORITHM (Adaptive Richardson's method).

Purpose. Execute RM and GMRES to solve a general linear-algebraic system.

Input. Matrix A; right side b; initial guess $x^{(0)}$; period k; m, where $m+1$ is the dimension of the Krylov subspace; and j, the number of computed eigenvalues. The parameter m may be set equal to j. The user must also provide a maximum number of passes and an error criterion to halt the algorithm [at steps 9(c) and 10]. An initial estimate of the convex hull of the spectrum of A could be provided if available. An expansion factor for the convex hull to be used in step 5 must also be provided.

Output. An approximate solution of $Ax = b$; an approximate convex hull of the spectrum of A.

Restriction. The period k is even, since the leapfrog variant is implemented. The dimension $m+1$ of the Krylov subspace must be large enough that j eigenvalues can be computed: $j \leqslant m$.

Notes.

1. A routine must be provided to evaluate the inner product $(f, g)_w = \sum_{i=1}^{M} f(\zeta_i)\overline{g(\zeta_i)}w(\zeta_i)m(\zeta_i)/M_w$.
2. A norm with no subscript denotes the standard Euclidcan norm of a complex vector, defined by $\|u\|^2 = \langle u, u \rangle$. The algorithm is adaptive: eigenvalue estimates are revised if necessary for convergence, based on a criterion which is tested in step 11.
3. The symbol C_H denotes an approximate convex hull of the spectrum of A.

Implementation details.

1. Only two eigenvalues are computed, except for the first pass, in which three are computed to obtain a convex hull with a nonvoid interior (if one exists).
2. The normal equations for the two LS problems of steps 3 and 8 are solved using LINPACK routines CGEFA (or CGECO) and CGESL.
3. The roots of the residual polynomial in step 3 are the eigenvalues of the companion matrix, and these are computed using the EISPACK routine CG.
4. the convex hull in step 4 is determined using "Graham's scan" algorithm [30, pp. 100 ff.].
5. The ζ_i in step 6 are determined as follows. Each edge between two successive vertices of the convex hull is subdivided into n points. The midpoints of these subdivisions are chosen as the ζ_i's. In our experiments, n was 5 and the points were so chosen that the subdivisions of a given edge were equal in length. Subdivisions of different edges would not necessarily be of equal length, however. Each $m(\zeta_i)$ was taken to be the length of the appropriate subsection. Note: If the convex hull is a line, as would occur if the eigenvalues were positive, more than five points will be necessary, since otherwise the degree of R_k could be at most five.
6. The LS problem of step 7 is solved by using a QR decomposition of F by means of LINPACK routines CQRDC and CQRSL.
7. An alternate version of step 8 would be to normalize the columns of the matrix, and then use appropriately modified α_i's.
8. The accuracy criterion of step 9(c) and step 10 is $\|r^{(k)}\| / \|r^{(0)}\| \leq$ *epsilon*, where *epsilon* is the input-error criterion provided by the user. The quantity $\|r^{(k)}\| / \|r^{(0)}\|$ will be called the *relative residual.*

Algorithm statement.

1. (Cf. [22, Chapter 13], step 1.) If there is an estimate of C_H, then continue; else set C_H equal to the null set and continue. Set *pass* $:= 1$. Set $r^{(0)} := b - Ax^{(0)}$.

2. (Cf. [22], step 2.) Set $s_i := A^i r^{(0)}$, for $i = 0, \ldots, m$.
3. (Cf. [22], step 3.) Solve the EGVL LS problem (10) for coefficients $c_0, \ldots, c_{j-1}$. In the case where $j = m = 2$ the eigenvalue normal equations to solve (10) are (with $\beta = 1/\|s_0\|^2$)

$$\beta\begin{bmatrix} s_0^* \\ s_1^* \end{bmatrix}\begin{bmatrix} s_0 & s_1 \end{bmatrix}\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 1 & \beta\langle s_1, s_0\rangle \\ & \beta\|s_1\|^2 \end{bmatrix}\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = -\begin{bmatrix} \beta\langle s_2, s_0\rangle \\ \beta\langle s_2, s_1\rangle \end{bmatrix}.$$

   Compute the roots, $\lambda_1, \ldots, \lambda_j$ of $c_0 + \cdots + c_{j-1}\lambda^{j-1} + \lambda^j = 0$ (to be used for approximate eigenvalues).
4. (Cf. [22], step 4.) Determine the convex hull of $C_H \cup \{\lambda_1, \ldots, \lambda_j\}$.
5. (New step not in [22].) Expand or contract the convex hull by an appropriate factor.
6. (Cf. [22], step 4.) Determine M points $\{\zeta_i\}$ and compute associated measures $m(\zeta_i)$ to define an inner product in accord with note 1 above.
7. (Cf. [22], step 5.) Compute roots $1/\tau_0, \ldots, 1/\tau_{k-1}$ of the optimal residual polynomial by solving the optimal-residual-polynomial LS problem as follows. Let $R_k(\zeta) = 1 + \Sigma_{i=1}^k \rho_i \zeta^i$. Note that $R_k(0) = 1$ (the definition of a residual polynomial). The discrete LS problem is to minimize $M_w\|R_k\|_w^2 = \|F(\bar{\rho}_1, \ldots, \bar{\rho}_k)^* - y\|^2$, where $F = (f_{ij})$ is the $M \times k$ matrix defined by $f_{ij} = \zeta_i^j[w(\zeta_i)m(\zeta_i)]^{1/2}$ and $y = (-[w(\zeta_1)m(\zeta_1)]^{1/2}, \ldots, -[w(\zeta_M)m(\zeta_M)]^{1/2})^*$. This may be solved by a standard LS method such as computing the QR decomposition of F.
8. (Cf. [22], step 6.) Solve the GMRES LS problem (13) for the parameters $\alpha_0, \ldots, \alpha_{j-1}$. In the case where $j = m = 2$ the normal equations to solve (13) are (with $\beta = 1/\|s_1\|^2$)

$$\beta\begin{bmatrix} s_1^* \\ s_2^* \end{bmatrix}\begin{bmatrix} s_1 & s_2 \end{bmatrix}\begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} 1 & \beta\langle s_2, s_1\rangle \\ & \beta\|s_2\|^2 \end{bmatrix}\begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \beta\langle s_0, s_1\rangle \\ \beta\langle s_0, s_2\rangle \end{bmatrix}.$$

   Set $x^{(0)} := x^{(m)}$, where $x^{(m)}$ is equal to the GMRES approximation as given by $x^{(m)} = x^{(0)} + \Sigma_{i=0}^{m-1}\alpha_i s_i$. If *pass* $= 1$, set $r^{(0)} := b - Ax^{(0)}$ and compute $\|r^{(0)}\|$ for possible use in step 9(c) and step 10.
9. (Cf. [22], step 7.) For $i := 2, \ldots, k$ by 2, DO:
   (a) (Cf. [22], step 7.1.) Set $\alpha := \tau_{i-2} + \tau_{i-1}$; set $\nu := \tau_{i-2}\tau_{i-1}$.
   (b) (Cf. [22], step 7.2.) Set $r^{(i-2)} := b - Ax^{(i-2)}$. (This may be omitted if either $i = 2$ and *pass* $= 1$ or $i = 2$ and this step follows immediately after step 11.) Set $t := Ar^{(i-2)}$.
   (c) (New step not in [22].) If an accuracy criterion is satisfied, exit.

(d) (Cf. [22], step 7.3.) Set $x^{(i)} := x^{(i-2)} + \alpha r^{(i-2)} - \nu t$.
(e) (Cf. [22], step 7.4.) Endfor.
(New step not in [22].) Set $r^{(k)} := b - Ax^{(k)}$.

10. (Cf. [22], step 8.) If either an accuracy criterion is satisfied or *pass* equals the maximum number allowed, exit; else set $x^{(0)} := x^{(k)}$, $r^{(0)} := r^{(k)}$, and $pass := pass + 1$.
11. (New alternative to [22], step 9.) If $(pass\text{-}2) \bmod 3$ equals 0, go to step 9; else go to step 2. Thus, new eigenvalue estimates are computed and GMRES is executed in two of three RM cycles.

6. NUMERICAL EXPERIMENTS

We shall refer to our algorithm as (the) adaptive RM (algorithm) and from time to time write "the (adaptive RM) algorithm" although, of course, it is only one adaptive algorithm out of many.

We compare adaptive RM with CGS, GMRES(5),[5] and CGNR.

Some special comments are appropriate for CGNR. CGNR has a bad reputation because it transforms the original system to one for which the condition number is the square of the condition number of the original system. However, in our experiments we often see that it is the best method, a fact observed by others [14, 39, 26] (and privately by Manteuffel). The success of CGNR on many of our examples is due to small condition numbers, a fact that should not mislead the alert reader.

A serious difficulty with CGNR is that it requires multiplication of a vector by the Hermitian transpose of the system matrix, a programming inconvenience in some cases, and an impossibility in the important case of matrix-free computations, for example, in the solution of nonlinear problems in which there is no explicit representation of the Jacobian [5].

A comparison between algorithms is easily made by comparing the number of matvecs to satisfy a stopping criterion if this operation takes most of the work, which is reasonable to assume. In general, RM takes fewer

[5]GMRES(m) is a restarted method for which the dimension of the Krylov subspace is never allowed to exceed m. For the details on GMRES, see [34]. The efficiency of GMRES(m) depends in a complicated way on the dimension of the subspace and the nature of the matrix. Generally it requires fewer iterations if the dimension is larger, but the total work may increase. No attempt was made to experiment with these parameters. Since the dimension of the Krylov subspace affects storage, it cannot be too large, and some limit is necessary. We chose dimension 5 because that number corresponds to the reasonable storage requirements of the Manteuffel algorithm.

arithmetic operations than GMRES or CGS, excluding the matvecs, but this is a secondary effect. In most experiments, the stopping criterion was to reduce the relative residual to 10^{-4}. Experiments were run on a variety of computers, with some variation observed among different computers. We consider only the arithmetic performance of an algorithm and make no attempt to consider system performance, such as the megaflop rate on either vector or parallel processors. Results are summarized in a table at the end of this section. Recall that the *relative residual at step k* is defined to be $\|r_k\|/\|r_0\|$.

6.1. *Tridiagonal Normal Matrices*

Many tests were conducted on matrices for which the spectra lie within the boomerang-shaped region defined by the vertices 5, $1 \pm 4i$, $3 \pm 4i$, 7, 5. There was no difference in the performance of our algorithm on matrices of order 100×100 and those of order $10{,}000 \times 10{,}000$, which is consistent with the dependence, in general, of the algorithm on the distribution of the eigenvalues of the system matrix, rather than the order. With eight iteration parameters (and an expansion factor of 1.5), the algorithm reached a relative residual of 10^{-4} after four passes (i.e., steps 9, 10, and 11, and steps 2–8 if required). The total number of matvecs was 36.

The adaptive Chebyshev algorithm of Manteuffel [23] in the Chebycode implementation [3] for a 1000×1000 matrix took 202 steps (217 matvecs). This method uses an ellipse to enclose an approximation to the spectrum, rather than the convex hull. The result suggests the convex hull is superior.

A more interesting comparison is with either CGS or GMRES. For a 16×16 matrix, CGS took 10 steps (21 matvecs), and for a 1000×1000 matrix, 16 steps (33 matvecs). For a 16×16 matrix, GMRES(5) took 6 cycles[6] (30 matvecs), and for a 1000×1000 matrix, also 6 steps (30 matvecs).

The best method is CGNR. For a 16×16 matrix, CGNR took 7 steps (14 matvecs), and for a 1000×1000 matrix, 8 steps (16 matvecs). The rapid convergence of CGNR is explained by the small condition number of $A^H A$, which is $\frac{49}{17} = 7^2/|1+4i|^2$.

6.2. *Indefinite Matrices*

Woo Sung Chi of the Department of Electrical and Computer Engineering of the University of Illinois, Urbana provided two 132×132 complex sample matrices. The first is one whose eigenvalues lie on the real line segment between -3 and 5 and is derived from the discretization of a boundary-value problem. In this case, CGS failed. Adaptive RM achieved a

[6]A *cycle* means the computation of a dimension-5 Krylov subspace.

relative-residual-error tolerance of 10^{-4} after 13 passes (using an expansion factor of 1.25 and an initial Krylov subspace dimension of 4, subsequently reduced to 3) (142 matvecs). GMRES(5) achieved the same tolerance after 52 cycles (262 matvecs), and CGNR after 71 steps (142 matvecs).

For the second sample matrix, also derived from a boundary-value problem, adaptive RM, CGS, and CGNR all failed. GMRES(5) was converging very slowly and was stopped after 143 cycles (715 matvecs), having achieved a relative residual error of 0.2892.

For our third example, we constructed a tridiagonal matrix whose eigenvalues were contained in the square with vertices at $\pm 2 \pm 2i$ above the line $y = i$ and below $y = -i$. For a 144×144 matrix, adaptive RM converged in 216 passes (2804 matvecs), but CGS had not converged after 1000 cycles (2001 matvecs). GMRES(5) failed, while CGNR converged after 11 cycles (22 matvecs). The condition number of this matrix is $8 = |2+2i|^2 / |i|^2$.

Our last example of an indefinite matrix is a complex matrix arising from the finite-element discretization of a 3D electromagnetic-field wave-propagation problem, provided by W. Mitchell and John D'Angelo (cf. [24]), for which there are somewhat fewer than 7000 unknowns. With the weight function $w(z) = 1$ [if $\Re(z) > 0$] or 5 [if $\Re(z) < 0$], the best results for adaptive RM were with an expansion factor of 0.9, tolerance of 10^{-3}, initial Krylov-space dimension of 3, and subsequent dimensions of 2. The variation in the weight function from one half plane to the other seemed to enhance the effect of polynomial preconditioning by (we conjecture) generating a preconditioned system with eigenvalues in a half plane. We conjecture that a hull contraction was effective due to a concentration of eigenvalues in the interior; errors corresponding to eigenvalues on the boundary of the hull were then damped by the GMRES method. Figure 5 shows the convex hull generated by the algorithm (using the weight function mentioned). (Different weight functions resulted in slightly different computed convex hulls, each of which resembles the hull displayed in Figure 5.)

For the adaptive RM, convergence was achieved in 246 passes with 2456 matvecs. CGS convergence was erratic and was stopped at 5000 steps (10,002 matvecs). GMRES(5) was stopped at 4464 cycles (22,320 matvecs), while CGNR achieved convergence in 452 steps (907 matvecs). These experiments were run on a Cray-XMP.

6.3. *Preconditioned Elliptic Difference Matrices*

We used a partial-difference test matrix described by H. Elman in [11] (see [12, p. 850] with $\gamma = 50$; also see [37]). There are two cases according as to whether the ILU or MILU preconditioning is used.

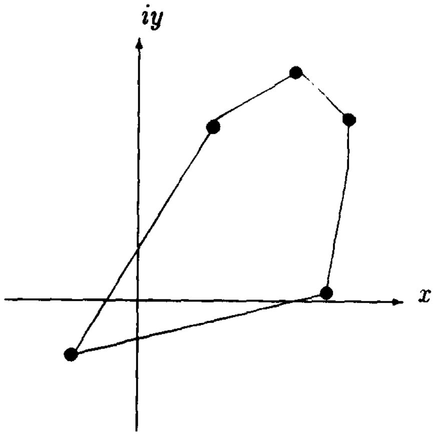

FIG. 5. Computed convex hull of the Mitchell-D'Angelo test matrix. Extremum eigenvalues are $-24.9-20.8i$, $71.3+2.96i$, $78.5+68.5i$, $58.7+87.2i$, and $27.7+66.1i$.

If ILU was used, and the matrix was 6400×6400, convergence for adaptive RM was reached in 3 passes (34 matvecs) with an explanation factor of 0.9, and a desired tolerance of 10^{-4}. CGS took 26 steps (53 matvecs). GMRES(5) converged in 6 cycles (30 matvecs). For smaller test matrices, CGS converged with fewer matvecs than our algorithm.

If we used the MILU preconditioner, then for the same order matrix, convergence of adaptive RM was reached in 3 passes (36 matvecs) with an expansion factor of 1.0. Here CGS took 25 steps (51 matvecs), and GMRES(5) took 4 cycles (20 matvecs).

For MILU preconditioning, CGS performance was smooth as the number of unknowns varied. However, for ILU preconditioning, CGS convergence was erratic (on a Pyramid minicomputer at the University of Illinois).

Comparisons with CGNR were reported in [11, pp. 150–152]. (The matrix size was not 6400×6400, but the results remain valid.) For these problems CGNR was not efficient.

6.4. *Summary of Experiments*

The numerical experiments are summarized in Table 1. In the table:

1. "Fail" indicates that the method was stopped prematurely due to divergence of the relative residual. In the case of CGS, failure means an attempted division by zero.

TABLE 1

SUMMARY OF DIFFERENT TEST MATRICES AND THE NUMBER OF MATVECS NEEDED TO ACHIEVE THE DESIRED TOLERANCE

Matrix	Ad. RM	CGS	GMRES(5)	CGNR
Boom. 16	36	21	30	14
Boom. 1000	36	33[a]	30	16
Chi no. 1	142	Fail	262	142
Chi no. 2	Fail	Fail	> 700	Fail
Square	2804	> 2001	Fail	22
Mit.-D'Ang.	2456	> 10,002	> 22,320	907
Elm. ILU 6400	34	53	30	—
Elm. MILU 6400	36	51	20	—

[a]A representative number of matvecs. The same code on a different machine led to more than 200 matvecs with no convergence. See text for further notes.

2. "—" indicates that the method was not tried on that test matrix.
3. "$> n$" indicates that the method was converging either slowly or very erratically and was halted at n matvecs.

7. OPTIONAL FORMULATIONS

In this section we shall outline a variant of the power method and a variant of the minimum residual step.

7.1. *Eigenvalues from the Shifted Matrix*

Let d be the center of the convex hull, and if, upon starting the algorithm no convex hull exists, let $d = 0$. In place of basing the power method on A, it may be an advantage to base it on $A - d$, to avoid biasing the large (in magnitude) eigenvalues of A. The Krylov subspace is unchanged, but the basis used for the computations is new:

$$V_{m+1} = \operatorname{span}\{r^{(0)}, \dots, (A-d)^m r^{(0)}\} \equiv \operatorname{span}\{r^{(0)}, \dots, A^m r^{(0)}\}.$$

Let $s_i = (A-d)^i r^{(0)}$. The eigenvalue LS problem is the same as before (Section 2.3), with the distinction, of course, that eigenvalues of $A - d$ are being approximated rather than eigenvalues of A. (The suggestion to use $A - d$ is due to John Castor [6].)

7.2. *The Minimum-Residual Computation*

A different Krylov subspace alters the formula for the GMRES approximation. The initial guess $x^{(0)}$ will be used in combination with the first m vectors. We shall describe this as well as a technique suggested by J. Grcar [18]. We believe it has potential, and therefore include it along with the modification necessary to employ the Krylov subspace generated by $A - d$.

Grcar's suggestion is that $x^{(m)}$ be of the form

$$x^{(m)} = \hat{\alpha}_0 x^{(0)} + \sum_{i=0}^{m-1} \alpha_i s_i.$$

The relation $As_{i-1} = s_i + ds_{i-1}$, $i \geqslant 1$, means that

$$A(x - x^{(m)}) = b - \delta_0 Ax^{(0)} - \delta_1 ds_0 - \sum_{i=0}^{m-1} \delta_{i+1} ds_i,$$

where $\delta_0 = \hat{\alpha}_0$, $\delta_1 = \alpha_0$, and for $i > 1$, $\delta_i = \alpha_i d + \alpha_{i-1}$.

The GMRES LS problem

$$\|A(x - x^{(m)})\| = \text{minimum}$$

yields, in the case $j = m = 2$, the normal equations

$$\begin{bmatrix} (Ax^{(0)})^* \\ s_1^* \\ s_2^* \end{bmatrix} \begin{bmatrix} Ax^{(0)} s_1 s_2 \end{bmatrix} \begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \end{bmatrix}$$

$$= \begin{bmatrix} \|Ax^{(0)}\|^2 & \langle s_1, Ax^{(0)} \rangle & \langle s_2, Ax^{(0)} \rangle \\ & \|s_1\|^2 & \langle s_2, s_1 \rangle \\ & & \|s_2\|^2 \end{bmatrix} \begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \end{bmatrix}$$

$$= \begin{bmatrix} \langle b, Ax^{(0)} \rangle \\ \langle b, s_1 \rangle \\ \langle b, s_2 \rangle \end{bmatrix}.$$

8. SUMMARY

We have presented an adaptive algorithm for the solution of linear-algebraic equations, which is a hybrid combination of GMRES and RM. An advantage of RM is that it allows matrix pipelining, and so GMRES, an important method also, receives short shrift. Other approaches to computing with RM have been recently presented, and we compare the mathematical basis for our approach with one of these. However, the advantages of matrix pipelining are independent of the method to obtain RM parameters. Numerical experiments compared the algorithm with CGS, GMRES, and CGNR, but the issue of matrix pipelining was outside the scope of these experiments. We should stress that our algorithm includes several user controlled features that affect performance, which means it is not automatic in the way that either CGS or CGNR is.

Support from the National Science Foundation grant DMS 87 03226 is gratefully acknowledged. The Mitchell-D'Angelo matrix experiments were run on a Cray-X-MP at the National Center for Supercomputing Applications at the University of Illinois, Champaign, Illinois. We thank W. Mitchell for providing the matrix. Computations were also performed at the Advanced Computing Research Facility, Mathematics and Computer Science Division, Argonne National Laboratory, Illinois. We appreciate the corrections and comments of Martin Gutknecht as well as those of the referee. We are indebted to Professor Walter Gander of the Institut für Wissenschaftliches Rechnen, Departement für Informatik, Eidgenössische Technische Hochschule, Zürich, for his hospitality.

REFERENCES

1 R. S. Anderssen and G. H. Golub, Richardson's Non-stationary Matrix Iterative Procedure, CS-72-304, Computer Science Dept., Stanford Univ., 1972.

2 S. F. Ashby, T. A. Manteuffel, and P. E. Saylor, Adaptive polynomial preconditioning for indefinite linear systems, *BIT* 29:12:583–609 (1989).

3 S. F. Ashby, CHEBYCODE: A FORTRAN Implementation of Manteuffel's Adaptive Chebyshev Algorithm, Research Report UIUCDCS-R-85-1203, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, 1985.

4 O. Axelsson, A survey of preconditioned iterative methods for linear systems of algebraic equations, *BIT* 25:166–187 (1985).

5 P. N. Brown and A. C. Hindmarsh, Matrix-free methods for stiff systems of ODEs, *SIAM J. Numer. Anal.* 23:610–683 (1986).

6 J. Castor, Private communication, Lawrence Livermore National Lab., Livermore, Calif., 1988.

7 A. Chronopoulos, A Class of Parallel Iterative Methods Implemented on Multiprocessors, Research Report UIUCDCS-R-86-1267, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Nov. 1986.

8 A. Chronopoulos and C. W. Gear, s-step iterative methods for symmetric linear systems, *J. Comput. Appl. Math.* 25:153–168 (1989).

9 A. Chronopoulos and C. W. Gear, On the efficient implementation of preconditioned s-step conjugate gradient methods on multiprocessors with memory hierarchy, *Parallel Comput.* 11:37–52 (1989).

10 M. Eiermann, R. S. Varga, and W. Niethammer, Iterationsverfahren für nichtsymmetrische Gleichungssysteme und Approximationsmethoden im Komplexen, *Jahresber. Deutsch. Math.-Verein* 89:1–32 (1987).

11 H. C. Elman, Iterative Methods for Large, Sparse Nonsymmetric Systems of Linear Equations, Ph.D. Thesis, Research Report 229, Dept. of Computer Science, Yale Univ., New Haven, Conn., 1982.

12 H. C. Elman, Y. Saad, and P. E. Saylor, A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations, *SIAM J. Sci. Statist. Comput.* 7:3:840–855 (1985).

13 H. C. Elman and R. L. Streit, Polynomial Iteration for Nonsymmetric Indefinite Linear Systems, Research Report 380, Dept. of Computer Science, Yale Univ., 1985.

14 B. Fischer and L. Reichel, A stable Richardson iteration method for complex linear systems, *Numer Math.* 54:225–241 (1988).

15 R. Fletcher, Conjugate gradient methods for indefinite systems, in: *Lecture Notes in Mathematics* 506, Springer-Verlag, Berlin, 1976, pp. 76–89.

16 G. H. Golub and M. L. Overton, The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems, *Numer. Math.* 53:571–593 (1988).

17 G. H. Golub and R. Kannan, Convergence of a two-stage Richardson process for nonlinear equations, *BIT* 26:209–216 (1986).

18 J. F. Grcar, Operator Coefficient Methods for Linear Equations, Research Report SAND89-8691, Sandia National Labs., Albuquerque, N. Mex., and Livermore, Calif., Nov. 1989.

19 M. H. Gutknecht, Iterative methods for linear systems of equations designed via complex rational approximations, *J. Approx. Theory* 89:1–4 (1989).

20 M. H. Gutknecht, Stationary and almost stationary iterative (k, l)-step methods for linear and nonlinear systems of equations, *Numer. Math.* 56:179–213 (1989).

21 L. A. Hageman and D. M. Young, *Applied Iterative Methods*, Academic, New York, 1981.

22 D. R. Kincaid and L. J. Hayes (Eds.), *Iterative Methods for Large Linear Systems*, Academic, Boston, 1990.

23 T. A. Manteuffel, Adaptive procedure for estimating parameters for the nonsymmetric Tchebyshev iteration, *Numer. Math.* 31:183–208 (1978).

24 W. F. Mitchell and J. D'Angelo, A nonsymmetric non-Hermitian complex matrix, in *Proceedings of the Copper Mountain Conference on Iterative Methods*, 1–5 Apr. 1990, Book 3.

25 P. D. Meyer, A. J. Valocchi, S. F. Ashby, and P. E. Saylor, A numerical investigation of the conjugate gradient method as applied to three-dimensional groundwater flow problems in randomly heterogeneous porous media, *Water Resources Res.* 26(6):1440–1446 (1989).

26 N. M. Nachtigal, L. Reichel, and L. N. Trefethen, A hybrid GMRES algorithm for nonsymmetric linear systems, in *Proceedings of the Copper Mountain Conference on Iterative Methods*, 1–5 Apr. 1990, Book 3.

27 D. M. Young, A historical review of iterative methods, in *A History of Scientific Computing* (S. G. Nash, Ed.), ACM Press (Addison-Wesley), New York, 1990.

28 G. Opfer and G. Schober, Richardson's iteration for nonsymmetric matrices, *Linear Algebra Appl.* 58:343–361 (1984).

29 B. N. Parsons, General k-part stationary iterative solutions to linear systems, *SIAM J. Numer Anal.* 24(1):188–198 (1984).

30 F. P. Preparata and M. I. Shamos, *Computational Geometry, an Introduction*, Springer-Verlag, New York, 1985.

31 L. Reichel, Polynomials by conformal mapping for the Richardson iteration method for complex linear systems, *SIAM J. Numer. Anal.* 25(6):1359–1368 (1988).

32 L. Reichel, The application of Leja points to Richardson iteration and polynomial preconditioning, *Linear Algebra Appl.*, to appear.

33 Y. Saad, Practical use of polynomial preconditionings for the conjugate gradient method, *SIAM J. Sci. Statist. Comput.* 7(3):865–881 (Oct. 1985).

34 Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* 7(3):856–869 (1986).

35 P. E. Saylor, Leapfrog variants of iterative methods for linear algebraic equations, *J. Comput. Appl. Math.* 24:169–193 (1988).

36 P. E. Saylor and D. C. Smolarski, Computing the roots of complex orthogonal and kernel polynomials, *SIAM J. Sci. Statist. Comput.* 9(1):1–13 (1988).

37 D. C. Smolarski and P. E. Saylor, An optimum iterative method for solving any linear system with a square matrix, *BIT* 28:163–178 (1988).

38 P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* 10(1):36–52 (1989).

39 H. Tal-Ezer, Polynomial approximation of functions of matrices and applications, *J. Sci. Comput.* 4:25–60 (1990).

40 H. Tal-Ezer, Private communication, 1987.

41 H. van der Vorst, Private communication, Urbana, Ill., 1981.

42 D. M. Young, On Richardson's method for solving linear systems with positive definite matrices, *J. Math. Phys.* 32:243–255 (1954).

Received 8 May 1990; final manuscript accepted 9 December 1990