SIAM REVIEW
Vol. 29, No. 2, June 1987

002

EXTRAPOLATION METHODS FOR VECTOR SEQUENCES*

DAVID A. SMITH†, WILLIAM F. FORD‡ AND AVRAM SIDI§

Abstract. This is an expository paper that describes and compares five methods for extrapolating to the limit (or anti-limit) of a vector sequence without explicit knowledge of the sequence generator. The methods are the minimal polynomial extrapolation (MPE) studied by Cabay and Jackson, Mešina, and Skelboe; the reduced rank extrapolation (RRE) of Eddy (which we show to be equivalent to Mešina's version of MPE); the vector and scalar versions of the epsilon algorithm (VEA, SEA) introduced by Wynn and extended by Brezinski and Gekeler; and the topological epsilon algorithm (TEA) of Brezinski. We cover the derivation and error analysis of iterated versions of the algorithms, as applied to both linear and nonlinear problems, and we show why these versions tend to converge quadratically. We also present samples from extensive numerical testing that has led us to the following conclusions: (a) TEA, in spite of its role as a theoretical link between the polynomial-type and the epsilon-type methods, has no practical application; (b) MPE is at least as good as RRE, and VEA at least as good as SEA, in almost all situations; (c) there are circumstances in which either MPE or VEA is superior to the other.

Key words. minimal polynomial extrapolation, reduced rank extrapolation, vector epsilon algorithm, scalar epsilon algorithm, topological epsilon algorithm, iterative methods, quadratic convergence

AMS(MOS) subject classifications. 65-02, 65B05, 65F10, 65H10

1. Introduction.

The purpose of this paper is to derive, describe, and compare five extrapolation methods for accelerating convergence of vector sequences or transforming divergent vector sequences to convergent ones. These methods have in common the property that they require no explicit knowledge of the "sequence generator" but are computed directly from the terms of the sequence. (In particular, if $\mathbf{x}_{n+1}=F(\mathbf{x}_n)$, no derivatives of F are computed.) Furthermore, when applied iteratively to a sequence generated nonlinearly, they tend to converge quadratically.

The methods studied belong to two families: *polynomial methods* and *epsilon algorithms*. The epsilon algorithms are old enough to be called "classical" in this field, dating to the mid-1950's for scalar sequences and to the early 1960's for vector sequences. We consider the *scalar epsilon algorithm* (SEA) and *vector epsilon algorithm* (VEA) introduced by Wynn [36] and the more recent *topological epsilon algorithm* (TEA) of Brezinski [8]. Polynomial extrapolation methods that are linear in the terms of the sequence, such as Chebyshev acceleration, have at least as long a history, but we consider only methods that are nonlinear, in that the coefficients of the extrapolating polynomials are functions of the terms of the sequence. The *minimal polynomial extrapolation* (MPE) and *reduced rank extrapolation* (RRE) are adapted from the works of Cabay and Jackson [12], Eddy [13], [14], Mešina [26], and Skelboe [31].

All five methods are based on the idea of *exact* solution for a fixed point in the case of a *linear* generator F, but without the equivalent of solving a system of equations in dimension N, the dimension of the vector space. The epsilon algorithms are computed recursively with a triangular array (of vectors) in which each entry is computed

*Received by the editors February 9, 1984; accepted for publication (in revised form) October 22, 1985.

†Department of Mathematics, Duke University, Durham, North Carolina 27706. The work of this author was supported by National Aeronautics and Space Administration grant NSG 3160.

‡Mathematical Analysis Section, NASA Lewis Research Center, Cleveland, Ohio 44135.

§Computer Science Department, Technion-Israel Institute of Technology, Haifa 3200, Israel. This work was done while the author was a National Research Council Research Associate at the NASA Lewis Research Center, Cleveland, Ohio.

from three earlier ones by simple arithmetic (no matrix inversions), and the exact solution is found as the leading entry in the $2k$th column, where k is the "essential degree" of the problem. We always have $k \leqq N$, and sometimes k is much smaller than N. The polynomial algorithms find the exact solution as a weighted average of $k+1$ terms of the sequence (same k), where the k independent weights are found by solving a linear system of that size.

In practice k is not known, but all of the methods may be applied with a possibly much smaller "effective degree," which is essentially the number of "dominant" eigenvalues of the sequence generator. Thus, even in the linear case the extrapolations are approximate, which leads to the need for iteration ("cycling"). If the sequence generator F is not linear, but has a Taylor expansion in which the linear part dominates (in a suitably small neighborhood of a fixed point), then the methods may still be applied. The transformed sequence obtained by cycling tends to converge quadratically, because small deviations from linearity in the initial data (given sequence) are transmitted linearly through the extrapolation.

In the next two sections, we derive MPE and RRE and prove that they give the exact solution for the right degree k. In §4, we present Brezinski's generalization [8] of the Shanks–Schmidt transform. The original transform led from systems of equations to Wynn's formulation [35] of the epsilon algorithm for scalar sequences. The generalized transform leads from systems of equations to TEA, and from there we make the necessary connections with SEA and VEA in §5. The exactness result for TEA is a consequence of the derivation (an intricate computation with determinants). For VEA it is a difficult theorem of McLeod [25], which has an extra hypothesis that remained a challenging puzzle until just recently, when Graves-Morris [19] approached the problem from a fresh point of view and gave the first satisfactory proof.

In §§6, 7, and 8 we extend the algorithms to the nonlinear case by cycling, sketch the error analysis for MPE and VEA (along the lines of Skelboe [31]), and discuss the theoretical support for quadratic convergence. Section 9 treats strategies for practical implementation of the methods. So far as we can tell, there is *no* practical implementation for TEA; its primary role seems to be as a theoretical "bridge" between the two families of methods. Section 10 explores alternate interpretations of these methods, additional relationships they have to each other and to other vector extrapolation methods (including Chebyshev and conjugate gradient), and further historical and bibliographical details. The paper closes with some numerical examples that illustrate the relative merits of the methods in various circumstances.

This paper is expository in nature. Our contribution to this subject consists of the analysis and synthesis of the work of others. Elsewhere [30] we present original work that includes introduction of a more general family of methods and a complete error analysis of these methods for linear problems without cycling, i.e. successive application of the extrapolation along the base sequence.

While this paper was in the refereeing process we learned of the work of Professor Jean Beuneu [2]. His work provides another way of studying in a unified context the methods discussed here.

2. Minimal polynomial extrapolation. Suppose a sequence of vectors in real or complex N-space is generated linearly from a starting point $\mathbf{x}_0$

$$\mathbf{x}_{j+1} = A\mathbf{x}_j + \mathbf{b}, \qquad j = 0, 1, 2, \cdots, \tag{2.1}$$

where A is a fixed matrix and $\mathbf{b}$ is a fixed vector. We do not assume that either A or $\mathbf{b}$ is known; only the sequence $\{\mathbf{x}_j\}$ or a means of generating it is given. We do assume that

1 is not an eigenvalue of A, so the iteration (2.1) has a unique fixed point

$$\mathbf{s} = A\mathbf{s} + \mathbf{b},$$

namely,

$$\mathbf{s} = (I - A)^{-1}\mathbf{b}. \tag{2.2}$$

If $|\lambda| < 1$ for every eigenvalue λ of A, then

$$\mathbf{s} = \lim_{j \to \infty} \mathbf{x}_j;$$

if the sequence diverges, $\mathbf{s}$ is called the *anti-limit*, and we may still expect to determine $\mathbf{s}$ from a finite number of the terms of the sequence. Our objective is to do that from as few terms as possible, without requiring any additional information about A, and without inverting an $N \times N$ matrix. We will derive, study, and compare several related methods that satisfy those criteria. In a typical application, N may be quite large relative to the number of eigenvalues λ with $|\lambda|$ near 1 (causing slow convergence) or greater than 1 (usually causing divergence). If the vectors are, say, successive discrete approximations to the solution of a partial differential equation by a finite difference scheme, then the sequence generator may be known to be linear (or approximately so), but A itself is likely to be unknown and expensive to compute. (The vector $\mathbf{b}$ can always be computed by starting the iteration at $\mathbf{x}_0 = 0$.) Furthermore, even when convergent, such schemes often require many thousands of iterations for suitable accuracy, and the individual terms $\mathbf{x}_j$ may themselves be expensive to compute.

The extrapolation methods to be derived are all based on differences, and it will be convenient to have abbreviated notation for first and second differences of the vectors $\mathbf{x}_j$. We write

$$\mathbf{u}_j = \Delta\mathbf{x}_j = \mathbf{x}_{j+1} - \mathbf{x}_j, \tag{2.3}$$

$$\mathbf{v}_j = \Delta^2\mathbf{x}_j = \Delta\mathbf{u}_j = \mathbf{u}_{j+1} - \mathbf{u}_j. \tag{2.4}$$

For a fixed integer k (to be determined), we define $N \times k$ matrices whose columns are the vectors of differences

$$U \equiv U_k = [\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{k-1}], \tag{2.5}$$

$$V \equiv V_k = [\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_{k-1}]. \tag{2.6}$$

Note that

$$\mathbf{u}_{j+1} = A\mathbf{u}_j = A^{j+1}\mathbf{u}_0 \tag{2.7}$$

and

$$\mathbf{v}_j = (A - I)\mathbf{u}_j \tag{2.8}$$

for each j.

The first method to be considered calculates $\mathbf{s}$ as a weighted average of the $\mathbf{x}_j$'s, with weights determined by the coefficients of the *minimal polynomial* $P(\lambda)$ of A with respect to $\mathbf{u}_0$, i.e. the unique monic polynomial of least degree such that

$$P(A)\mathbf{u}_0 = 0. \tag{2.9}$$

For the present, we take k to be the degree of P. Then $k \leqq N$, and it may happen that k is much smaller than N. We write

$$P(\lambda) = \sum_{j=0}^{k} c_j \lambda^j, \qquad c_k = 1.$$

Then it follows from (2.7) and (2.9) that

$$\sum_{j=0}^{k} c_j \mathbf{u}_j = 0, \tag{2.10}$$

so the vector $\mathbf{c} = (c_0, c_1, \cdots, c_{k-1})^t$ of unknown coefficients of P is a solution of the system of equations

$$U\mathbf{c} = -\mathbf{u}_k. \tag{2.11}$$

If $k < N$, the system (2.11) has more equations than unknowns, but consistency has just been demonstrated, and the unique solution may be written as

$$\mathbf{c} = -U^+ \mathbf{u}_k, \tag{2.12}$$

where U^+ is the pseudoinverse (or Moore–Penrose generalized inverse) of U (see [23] or [32]). In principle, computation of U^+ calls for inversion of a $k \times k$ matrix; we will discuss practical computation of $\mathbf{c}$ later.

Before stating the extrapolation formula for obtaining $\mathbf{s}$ from the $\mathbf{x}_j$'s, we observe that $\mathbf{c}$ gives us the coefficients for the minimal polynomials of A with respect to *all* of the vectors $\mathbf{u}_j$, and also with respect to all the error vectors $\mathbf{x}_j - \mathbf{s}$. We write $P(\lambda) = \lambda^r Q(\lambda)$, where r is the multiplicity of λ as a factor of P. Thus, $c_0 = c_1 = \cdots = c_{r-1} = 0$, and

$$Q(\lambda) = \sum_{j=r}^{k} c_j \lambda^{j-r}, \quad c_r \neq 0, \quad c_k = 1.$$

Since P is a factor of the characteristic polynomial of A, r is $\leqq$ the multiplicity of 0 as an eigenvalue. Often r will be 0, of course, for example, if A is nonsingular.

LEMMA 1. *The minimal polynomial $P_j(\lambda)$ of A with respect to $\mathbf{u}_j$ is $\lambda^{r-j} Q(\lambda)$ for $j = 0, 1, \cdots, r$, and is $Q(\lambda)$ for $j > r$. The minimal polynomial of A with respect to $\mathbf{x}_j - \mathbf{s}$ is the same as that for $\mathbf{u}_j$, for every j.*

Proof. For $j \leqq r$,

$$A^{r-j} Q(A) \mathbf{u}_j = A^{r-j} Q(A) A^j \mathbf{u}_0 = P(A) \mathbf{u}_0 = \mathbf{0},$$

so $P_j(\lambda)$ divides $\lambda^{r-j} Q(\lambda)$. On the other hand,

$$\mathbf{0} = P_j(A) \mathbf{u}_j = A^j P_j(A) \mathbf{u}_0,$$

so $P(\lambda) = \lambda^r Q(\lambda)$ divides $\lambda^j P_j(\lambda)$. It follows that $P_j(\lambda) = \lambda^{r-j} Q(\lambda)$, since both polynomials are monic.

For $j > r$, a similar computation shows that P_j divides Q, and Q divides $\lambda^{j-r} P_j$. Since λ is not a factor of Q, it follows that $P_j = Q$.

To prove the second assertion, we observe that

$$(A - I)(\mathbf{x}_j - \mathbf{s}) = \mathbf{u}_j, \tag{2.13}$$

which follows immediately from the definitions. Since $A-I$ is assumed to be invertible and commutes with polynomials in A, it follows that the minimal polynomials with respect to $\mathbf{u}_j$ and $\mathbf{x}_j-\mathbf{s}$ must be the same. □

THEOREM 1. *For any $k+1$ consecutive terms of the sequence, say $\mathbf{x}_m, \mathbf{x}_{m+1}, \cdots, \mathbf{x}_{m+k}$, we have*

$$\sum_{j=0}^{k} c_j \mathbf{x}_{m+j} = \left(\sum_{j=0}^{k} c_j \right) \mathbf{s}. \tag{2.14}$$

Proof. We first solve the recursion (2.1) explicitly to express $\mathbf{x}_{m+j}$ in terms of $\mathbf{x}_m$, using (2.2)

$$\begin{aligned} \mathbf{x}_{m+j} &= A^j \mathbf{x}_m + (I + A + \cdots + A^{j-1})\mathbf{b} \\ &= A^j \mathbf{x}_m + (I - A^j)(I - A)^{-1}\mathbf{b} \\ &= A^j \mathbf{x}_m + (I - A^j)\mathbf{s} \\ &= A^j(\mathbf{x}_m - \mathbf{s}) + \mathbf{s}. \end{aligned}$$

From Lemma 1, we have

$$\begin{aligned} \mathbf{0} &= \sum_{j=0}^{k} c_j A^j (\mathbf{x}_m - \mathbf{s}) \\ &= \sum_{j=0}^{k} c_j (\mathbf{x}_{m+j} - \mathbf{s}) \\ &= \sum_{j=0}^{k} c_j \mathbf{x}_{m+j} - \left(\sum_{j=0}^{k} c_j \right) \mathbf{s}. \end{aligned}$$

□

Note that $\mathbf{s}$ may be computed exactly from (9), since $\Sigma c_j = P(1) \neq 0$, because we assumed that 1 is not an eigenvalue of A. Note also that if $P(0)=0$, r terms on each side of (2.14) are zero, where r is the multiplicity of 0 as a root of P. If r is known or suspected to be positive, there is some advantage in starting the extrapolation process at $\mathbf{x}_m$ for $m>0$, preferably $m=r$, since each step away from $\mathbf{x}_0$ reduces the size of the matrix to be inverted in (2.12) by one. For simplicity, we will continue to refer to the starting point for this extrapolation (and the others to be derived) as $\mathbf{x}_0$, but it is to be understood that each extrapolation may be applied after some number of steps with the sequence generator. Finally, note that the determination of $\mathbf{s}$ from (2.14) actually requires $k+2$ terms of the sequence, since $k+1$ differences $\mathbf{u}_j$ are needed to find $\mathbf{c}$. We summarize the algorithm as follows:

Minimal Polynomial Extrapolation (MPE). Given a sequence generator of the form (2.1) and a starting point $\mathbf{x}_0$,

(a) generate $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{k+1}$;
(b) compute U, $\mathbf{u}_k$ as in (2.3) and (2.5);
(c) compute $\mathbf{c}$ as in (2.11), (2.12), and set $c_k=1$;
(d) compute $\mathbf{s}$ from (2.14).

The form of MPE just stated is somewhat simpler than that given by Cabay and Jackson [12]. We now derive their formulas to establish a connection with the method to be considered in the next section.

Define a polynomial $R(\lambda)$ by

$$R(\lambda)=\sum_{j=0}^{k-1}\left(\sum_{i=j+1}^{k}c_i\right)\lambda^j\equiv\sum_{j=0}^{k-1}b_j\lambda^j. \tag{2.15}$$

Then

$$\begin{aligned}(1-\lambda)R(\lambda)&=\sum_{j=0}^{k+1}\sum_{i=j+1}^{k}c_i(\lambda^j-\lambda^{j+1})\\&=\sum_{i=1}^{k}\sum_{j=0}^{i-1}c_i(\lambda^j-\lambda^{j+1})\\&=\sum_{i=1}^{k}c_i(1-\lambda^i)\\&=P(1)-P(\lambda).\end{aligned}$$

Thus

$$(I-A)R(A)\mathbf{u}_0=[P(I)-P(A)]\mathbf{u}_0=P(1)\mathbf{u}_0.$$

From (2.13), we have

$$\mathbf{s}-\mathbf{x}_0=(I-A)^{-1}\mathbf{u}_0=P(1)^{-1}R(A)\mathbf{u}_0=P(1)^{-1}\sum_{j=0}^{k-1}b_j\mathbf{u}_j.$$

Thus formula (2.14) in MPE may be replaced by

$$\mathbf{s}=\mathbf{x}_0-\left(\sum_{j=0}^{k-1}b_j\mathbf{u}_j\right)\Big/\left(\sum_{j=0}^{k}c_j\right), \tag{2.16}$$

where the coefficients b_j are defined by (2.15).

3. Reduced rank extrapolation. If k is taken to be N in (2.5) and (2.6), and if V is invertible, then it follows from (2.8) that

$$(I-A)^{-1}=-UV^{-1},$$

and hence from (2.13) that

$$\mathbf{s}=\mathbf{x}_j-UV^{-1}\mathbf{u}_j \tag{3.1}$$

for any j, in particular for $j=0$. This is a "full rank extrapolation"; it requires inversion of a rank N matrix, and its only advantage over (2.2) is that A and $\mathbf{b}$ still need not be known explicitly.

However, (3.1) may be rewritten as a pair of equations

$$\mathbf{s}=\mathbf{x}_0+U\boldsymbol{\xi}, \tag{3.2}$$

$$\mathbf{0}=\mathbf{u}_0+V\boldsymbol{\xi}. \tag{3.3}$$

Now we may ask if there is a simultaneous solution ξ of (3.2) and (3.3) when U and V have k columns, where k may be less than N. The answer is "yes," and for $k=\deg P$ as in the previous section. In fact, if we set

$$\xi_j=b_j/P(1), \qquad 0\leqq j\leqq k-1,$$

then it follows from (2.16) that ξ satisfies (3.2), and from (2.8) and (2.13) we have

$$V\xi=(A-I)U\xi=(A-I)(\mathbf{s}-\mathbf{x}_0)=-\mathbf{u}_0,$$

so (3.3) is also satisfied.

Now ξ may be computed directly from (3.3), consistency having been demonstrated, and the result substituted in (3.2) to get the "reduced rank" equivalent of (3.1)

$$\mathbf{s}=\mathbf{x}_0-UV^{+}\mathbf{u}_0. \tag{3.4}$$

Note that this requires the equivalent of an inversion of a $k\times k$ matrix, but not the same one that appears in MPE. We summarize the algorithm as follows:

Reduced Rank Extrapolation (RRE). Given a sequence generator of the form (2.1) and a starting vector $\mathbf{x}_0$,

(a) generate $\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_{k+1}$;
(b) compute U, $\mathbf{u}_k$ as in (2.3) and (2.5);
(c) compute V as in (2.4) and (2.6);
(d) compute the generalized inverse V^{+};
(e) compute $\mathbf{s}$ from (3.4).

Note that exactly the same iterates are required for RRE and for MPE, in spite of the second differences used for the former; $\mathbf{v}_k$ is not used, so $\mathbf{x}_{k+2}$ is not needed. One might expect that second differences of a slowly convergent sequence would cause more numerical problems than first differences, and this is sometimes the case, as we shall see. Apart from these exceptional cases, however, MPE and RRE are quite similar in their numerical performance.

If we write ΔX for U and $\Delta^2 X$ for V (where X is a matrix whose columns are the iterates $\mathbf{x}_j$, and Δ is applied to columns), then (3.4) becomes

$$\mathbf{s}=\mathbf{x}_0-\Delta X(\Delta^2 X)^{+}\Delta\mathbf{x}_0.$$

In this form it is clear that RRE is a natural extension of Aitken's Δ^2 method [1] to vector problems.

4. The generalized Shanks–Schmidt transform. We see from Theorem 1 that the fixed point $\mathbf{s}$ can be expressed as a weighted average of any $k+1$ consecutive terms of the sequence $\{\mathbf{x}_j\}$, where k is the degree of the minimal polynomial P of A with respect to $\mathbf{x}_0-\mathbf{s}$

$$\sum_{j=0}^{k}\gamma_j\mathbf{x}_{m+j}=\mathbf{s}, \qquad m=0,1,2,\cdots, \tag{4.1}$$

$$\sum_{j=0}^{k}\gamma_j=1. \tag{4.2}$$

(Here $\gamma_j = c_j/P(1)$.) This does *not* mean that the extrapolation to $\mathbf{s}$ from $\{\mathbf{x}_j\}$ is being computed linearly, however, because of the implicit matrix inversion required for the computation of the c_j (hence also the γ_j), where the matrix entries are linear combinations of the components of the $\mathbf{x}_j$. All of the methods discussed here require nonlinear combinations of the vectors in the given sequence.

Each equation of the form (4.1) is equivalent to N scalar equations in the unknowns $\{\gamma_j\}$ and the components of $\mathbf{s}$, and there are perhaps an infinite number of different equations. We know that the entire system is consistent when the "right" k is used, but in practice we cannot expect to have a priori knowledge of k, so we can only hope to find approximate values of k, $\{\gamma_j\}$, and therefore $\mathbf{s}$, from our extrapolations, whether by MPE, RRE, or the methods to be described in this section and the next. Thus, we are often dealing with a selection of an appropriate number of linear equations taken from (or derived from) a large, inconsistent set. We consider now another way to solve the system (4.1)–(4.2), which will be exact for $k = \deg P$.

Take $k+1$ consecutive equations of the form (4.1), say with $n \leqq m \leqq n+k$, and eliminate $\mathbf{s}$ from these equations by taking differences (second equation minus first, etc.)

$$\sum_{j=0}^{k} \gamma_j \mathbf{u}_{m+j} = \mathbf{0}, \qquad m = n, n+1, \cdots, n+k-1. \tag{4.3}$$

Now let $\mathbf{y}$ be an arbitrary nonzero vector in N-space, and consider the scalar equations obtained by taking the inner product of $\mathbf{y}$ with each of the vectors in (4.3)

$$\sum_{j=0}^{k} \gamma_j (\mathbf{y} \cdot \mathbf{u}_{m+j}) = 0, \qquad m = n, n+1, \cdots, n+k-1. \tag{4.4}$$

The system consisting of (4.2) and (4.4) constitutes $k+1$ linear equations in the $k+1$ unknowns $\{\gamma_j\}$, which will have a unique solution if the appropriate determinant is nonzero, a modest restriction on the arbitrariness of $\mathbf{y}$. If we abbreviate $\mathbf{y} \cdot \mathbf{u}_{m+j}$ by z_{m+j}, that determinant is

$$\begin{vmatrix} 1 & 1 & \cdots & 1 \\ z_n & z_{n+1} & \cdots & z_{n+k} \\ z_{n+1} & z_{n+2} & \cdots & z_{n+k+1} \\ \cdots & \cdots & \cdots & \cdots \\ z_{n+k-1} & z_{n+k} & \cdots & z_{n+2k-1} \end{vmatrix}. \tag{4.5}$$

By subtracting adjacent columns (first from second, second from third, etc.) and expanding on the first row, we find that the determinant (4.5) of order $k+1$ is equal to the kth order determinant

$$\begin{vmatrix} \Delta z_n & \Delta z_{n+1} & \cdots & \Delta z_{n+k-1} \\ \Delta z_{n+1} & \Delta z_{n+2} & \cdots & \Delta z_{n+k} \\ \cdots & \cdots & \cdots & \cdots \\ \Delta z_{n+k-1} & \Delta z_{n+k} & \cdots & \Delta z_{n+2k-2} \end{vmatrix}, \tag{4.6}$$

which is the (classical) Hankel determinant $H_k^{(n)}(\Delta z_n)$, starting at the nth term of the scalar sequence $\{\Delta z_n\} = \{\mathbf{y} \cdot \mathbf{v}_n\}$.

Following Brezinski [9], we define a *generalized Hankel determinant* for an arbitrary vector sequence $\{\mathbf{w}_n\}$ with respect to a vector $\mathbf{y}$

$$\tilde{H}_{k+1}^{(n)}(\mathbf{w}_n)=\begin{vmatrix} \mathbf{w}_n & \mathbf{w}_{n+1} & \cdots & \mathbf{w}_{n+k} \\ \mathbf{y}\cdot\Delta\mathbf{w}_n & \mathbf{y}\cdot\Delta\mathbf{w}_{n+1} & \cdots & \mathbf{y}\cdot\Delta\mathbf{w}_{n+k} \\ \mathbf{y}\cdot\Delta\mathbf{w}_{n+1} & \mathbf{y}\cdot\Delta\mathbf{w}_{n+2} & \cdots & \mathbf{y}\cdot\Delta\mathbf{w}_{n+k+1} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{y}\cdot\Delta\mathbf{w}_{n+k-1} & \mathbf{y}\cdot\Delta\mathbf{w}_{n+k} & \cdots & \mathbf{y}\cdot\Delta\mathbf{w}_{n+2k-1} \end{vmatrix}$$

where the interpretation of "determinant" with a row of vector entries is the linear combination of that row formed by the usual expansion formula, with (scalar) coefficients that are ordinary kth order determinants. The inner product $\mathbf{y}\cdot\tilde{H}_{k+1}^{(n)}(\mathbf{w}_n)$ is an ordinary determinant, and by successively adding each row to the next, we may see it is a Hankel determinant. In fact,

$$\mathbf{y}\cdot\tilde{H}_{k+1}^{(n)}(\mathbf{w}_n)=H_{k+1}^{(n)}(\mathbf{y}\cdot\mathbf{w}_n). \tag{4.7}$$

Furthermore, in the scalar case (dimension $N=1$) (4.7) reduces to

$$\tilde{H}_{k+1}^{(n)}(w_n)=y^k H_{k+1}^{(n)}(w_n) \tag{4.8}$$

which shows that $\tilde{H}_{k+1}^{(n)}$ is a proper generalization to vector sequences of the Hankel determinant for scalar sequences.

Now if one solves (4.2), (4.4) for the γ_j's by Cramer's Rule and substitutes the result in (4.1), the solution for $\mathbf{s}$ may be written as

$$\mathbf{s}=\tilde{H}_{k+1}^{(n)}(\mathbf{x}_n)/H_k^{(n)}(\mathbf{y}\cdot\Delta^2\mathbf{x}_n). \tag{4.9}$$

Since k will usually not be known, we consider the right-hand members of (4.9) to be a double sequence of approximations to $\mathbf{s}$ and write

$$e_k(\mathbf{x}_n)=\tilde{H}_{k+1}^{(n)}(\mathbf{x}_n)/H_k^{(n)}(\mathbf{y}\cdot\Delta^2\mathbf{x}_n). \tag{4.10}$$

For historical reasons, we will call the sequence transformation defined by (4.10) the *generalized Shanks–Schmidt* (GSS) transform. In the scalar ($N=1$) case, we see from (4.8) with $y=1$ and the equivalence of (4.5) and (4.6) that (4.10) reduces to

$$e_k(x_n)=H_{k+1}^{(n)}(x_n)/H_k^{(n)}(\Delta^2 x_n) \tag{4.11}$$

$$=\frac{\begin{vmatrix} x_n & x_{n+1} & \cdots & x_{n+k} \\ \Delta x_n & \Delta x_{n+1} & \cdots & \Delta x_{n+k} \\ \cdots & \cdots & \cdots & \cdots \\ \Delta x_{n+k-1} & \Delta x_{n+k} & \cdots & \Delta x_{n+2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \Delta x_n & \Delta x_{n+1} & \cdots & \Delta x_{n+k} \\ \cdots & \cdots & \cdots & \cdots \\ \Delta x_{n+k-1} & \Delta x_{n+k} & \cdots & \Delta x_{n+2k-1} \end{vmatrix}},$$

which is the (classical) e-transform studied independently by Shanks [29] and Schmidt [28].

The e-transform (4.11) is known to give the exact limit or anti-limit s for any scalar sequence $\{x_n\}$ whose errors satisfy

$$x_n-s=\sum_{i=1}^{k} a_i\lambda_i^n, \tag{4.12}$$

and it has been found to give useful approximations to s if the true expression for the errors involves an infinite sum, i.e. infinitely many "transients" λ_i. If $k=1$ in (4.11), the e-transform reduces to Aitken's Δ^2 extrapolation, which is exact on geometric series, the case of $k=1$ in (4.12). Thus, like RRE, GSS is a natural generalization of Aitken's Δ^2 to vector sequences.

The vector analogue of condition (4.12) is

$$\mathbf{x}_n-\mathbf{s}=\sum_{i=1}^{k}\mathbf{a}_i\lambda_i^n, \tag{4.13}$$

where the $\mathbf{a}_i$ are fixed vectors and the λ_i fixed scalars. The subspace generated by $\mathbf{x}_0-\mathbf{s}$ (i.e. spanned by $\mathbf{x}_0-\mathbf{s}$, $A(\mathbf{x}_0-\mathbf{s})$, $A^2(\mathbf{x}_0-\mathbf{s}),\dots$) has dimension $k=\deg P$. Every error vector $\mathbf{x}_n-\mathbf{s}$ is in that subspace (see the proof of Theorem 1), and if A is nondefective on that subspace, then we have a decomposition of the form (4.13) by taking the λ_i to be eigenvalues of A and $\mathbf{a}_i$ to be the corresponding eigenvectors in a decomposition of $\mathbf{x}_0-\mathbf{s}$. In the defective case, the representation of $\mathbf{x}_n-\mathbf{s}$ in terms of generalized eigenvectors is more complicated, but we still know that GSS gives the exact result (4.9) if $k=\deg P$.

The GSS, as given by (4.10), would be of little value for approximating the fixed point $\mathbf{s}$ if it had to be evaluated by forming a linear combination of the $\mathbf{x}_j$'s whose coefficients are computed as ratios of (large) determinants. Indeed, even in the scalar case, the e-transform would have found little practical application if it were not for the recursive evaluation scheme, discovered by Wynn [35], called the *epsilon algorithm*, to which we now turn our attention.

5. The epsilon algorithm. With a remarkable burst of insight, Wynn [35] discovered, very soon after Shanks [29] published his paper on the sequence transformation (4.11), that the required ratio of determinants could be evaluated recursively for increasing k and n, without the use of determinants or matrix inversion. The result is the (scalar) *epsilon algorithm*, which is given by the formulas

$$\varepsilon_{-1}^{(n)}=0,\quad \varepsilon_0^{(n)}=x_n,\quad n=0,1,2,\cdots, \tag{5.1}$$

$$\varepsilon_{k+1}^{(n)}=\varepsilon_{k-1}^{(n+1)}+\left[\varepsilon_k^{(n+1)}-\varepsilon_k^{(n)}\right]^{-1},\qquad k,n=0,1,2,\cdots. \tag{5.2}$$

By working along diagonals on which $n+k$ is constant, (5.2) permits calculation of each entry of a triangular array in terms of previously calculated entries, with initial conditions given by (5.1).

THEOREM 2 (Wynn). *For each pair of nonnegative integers k and n, if the indicated quantities exist,*

$$\varepsilon_{2k}^{(n)}=e_k(x_n),\qquad \varepsilon_{2k+1}^{(n)}=1/e_k(\Delta x_n). \tag{5.3}$$

In particular, the even numbered columns of the ε array evaluate the Shanks–Schmidt transform (4.11). *Furthermore, if $\{x_n\}$ satisfies* (4.12) *for some k, then $\varepsilon_{2k}^{(n)}=s$ for every n.*

Once we see what has to be proved, a proof by induction is not difficult, but it requires several pages of determinantal identities and expansions, which we choose not to reproduce here. (See [35] or [9, pp. 44–47] for the details.)

The principal stumbling block in the extension of the epsilon algorithm to vector sequences is the appropriate interpretation of the inversion (reciprocal) in (5.2) when applied to a vector quantity. One possibility is to treat the sequence of ith components

of $\{\mathbf{x}_n\}$ as a separate scalar sequence for each i, so that "inverse" continues to mean "reciprocal." This algorithm, as applied to vector sequences, is called the *scalar epsilon algorithm* (SEA). Of course, this approach ignores the linkage between the scalar sequences in different components, and it also runs the risk that, in one or more components, the required reciprocals frequently will either fail to exist or be quite large numerically. (Singular rules have been devised to work around this latter problem, but we choose not to digress to discuss them here.) In spite of these obvious difficulties, SEA works quite well in a surprisingly large number of cases.

A connection between SEA and GSS may be made in the following way. If one chooses the arbitrary vector $\mathbf{y}$ in (4.10) to be the ith standard unit vector in N-space, and then ignores all but the ith component of the extrapolation, the result will be the ith component of the vector $\boldsymbol{\varepsilon}_{2k}^{(n)}$ computed by SEA.

Brezinski [8], [9, pp. 172–205] has shown how to make a stronger connection between the epsilon algorithm and GSS by an appropriate interpretation of "inverse" of a vector, i.e. how to solve the system (4.2), (4.4) to produce the extrapolations (4.10) by recursive formulas of the form (5.1), (5.2). Indeed, the possibility of such a connection was his reason for introducing GSS, which, as we have noted, is not an effective computational scheme in itself.

We define the *inverse* of an ordered pair $(\mathbf{a}, \mathbf{b})$ of vectors such that $\mathbf{a}\cdot\mathbf{b}\neq 0$ to be the ordered pair $(\mathbf{b}^{-1}, \mathbf{a}^{-1})$, where

$$\mathbf{b}^{-1}=\mathbf{a}/(\mathbf{b}\cdot\mathbf{a}), \qquad \mathbf{a}^{-1}=\mathbf{b}/(\mathbf{b}\cdot\mathbf{a}).$$

We call $\mathbf{a}^{-1}$ the *inverse of* $\mathbf{a}$ *with respect to* $\mathbf{b}$, and vice versa. Several properties of this concept of inversion are immediate.

$$(\mathbf{a}^{-1})^{-1}=\mathbf{a} \quad \text{(the second inversion with respect to } \mathbf{a}),$$
$$\mathbf{a}^{-1}\cdot\mathbf{a}=1,$$
$$\mathbf{a}^{-1}\cdot\mathbf{b}^{-1}=1/(\mathbf{b}\cdot\mathbf{a}).$$

Now we define a vector version of (5.2) with inversion of vectors interpreted in the following way: For even subscript $2k$, the inverse of $\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}$ is with respect to a fixed vector $\mathbf{y}$, arbitrary except for the restriction that all the required inverses should exist, i.e. $\mathbf{y}\cdot\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}\neq 0$ for each k and n. For odd subscript, the inversion is with respect to the $\mathbf{y}^{-1}$ determined at the previous step. Thus we have

$$\left[\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}\right]^{-1}=\mathbf{y}/\left(\mathbf{y}\cdot\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}\right),$$
$$\mathbf{y}^{-1}=\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}/\left(\mathbf{y}\cdot\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}\right),$$
$$\left[\Delta\boldsymbol{\varepsilon}_{2k+1}^{(n)}\right]^{-1}=\mathbf{y}^{-1}/\left(\Delta\boldsymbol{\varepsilon}_{2k+1}^{(n)}\cdot\mathbf{y}^{-1}\right)=\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}/\left(\Delta\boldsymbol{\varepsilon}_{2k+1}^{(n)}\cdot\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}\right).$$

This leads to the *topological epsilon algorithm* (TEA)

$$\boldsymbol{\varepsilon}_{-1}^{(n)}=\mathbf{0}, \quad \boldsymbol{\varepsilon}_{0}^{(n)}=\mathbf{x}_n, \quad n=0,1,2,\cdots, \tag{5.4}$$

$$\boldsymbol{\varepsilon}_{2k+1}^{(n)}=\boldsymbol{\varepsilon}_{2k-1}^{(n)}+\mathbf{y}/\left(\mathbf{y}\cdot\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}\right), \qquad k,n=0,1,2,\cdots, \tag{5.5}$$

$$\boldsymbol{\varepsilon}_{2k+2}^{(n)}=\boldsymbol{\varepsilon}_{2k}^{(n)}+\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}/\left(\Delta\boldsymbol{\varepsilon}_{2k+1}^{(n)}\cdot\Delta\boldsymbol{\varepsilon}_{2k}^{(n)}\right), \qquad k,n=0,1,2,\cdots. \tag{5.6}$$

The name derives from the fact that the entire development of the previous section and this one can be carried out in the context of a topological vector space E, with dot product by $\mathbf{y}$ replaced by the application of an arbitrary element of the topological dual

space E'. In this generality, the even and odd columns of the epsilon array are alternately in E and E', respectively, and inversion is defined on (most of) $E \times E'$.

THEOREM 3 (Brezinski). *If an epsilon table is generated from a sequence* $\{\mathbf{x}_n\}$ *by formulas* (5.4)–(5.6) *with a fixed vector* $\mathbf{y}$, *and if all the indicated quantities exist, then*

$$\varepsilon_{2k}^{(n)} = e_k(\mathbf{x}_n), \tag{5.7}$$

and

$$\varepsilon_{2k+1}^{(n)} = [e_k(\mathbf{u}_n)]^{-1} = \mathbf{y}/[\mathbf{y} \cdot e_k(\mathbf{u}_n)], \tag{5.8}$$

for $n, k = 0, 1, 2, \cdots$, *where* e_k *is given by* (4.10).

We may see that Theorem 3 is a generalization of Theorem 2 by considering the case of dimension $N = 1$, and indeed the proof of Theorem 3 is essentially the observation that the definitions of GSS and TEA have been properly constructed to permit Wynn's argument to be extended to generalized determinants. See [8] or [9] for the details.

Our primary purpose in describing TEA is to demonstrate the connecting link between "polynomial" type algorithms such as MPE and RRE and "epsilon" type algorithms. TEA solves equations (4.1)–(4.4) for $\mathbf{s}$ without the necessity of matrix inversion, but at a cost of generating almost twice as many terms of the sequence $\{\mathbf{x}_n\}$. However, TEA has not been notably successful as a numerical algorithm, and little is known about appropriate selection of the auxiliary vector $\mathbf{y}$, let alone optimal selection.

A much more successful extension of (5.1)–(5.2) to the vector case is that given by Wynn [36], who suggested interpretation of "inverse" as the *Samelson inverse*

$$\mathbf{w}^{-1} = \mathbf{w}/\|\mathbf{w}\|^2. \tag{5.9}$$

This is also the transpose of the Moore–Penrose generalized inverse of $\mathbf{w}$, considered as a matrix. Furthermore, it is a special case of Brezinski's notion of inverse of an ordered pair, with only pairs of the form $(\mathbf{w}, \overline{\mathbf{w}})$ considered. The vector version of (5.1)–(5.2) in this case is called the *vector epsilon algorithm* (VEA), and the recursive formulas take the form

$$\varepsilon_{-1}^{(n)} = \mathbf{0}, \quad \varepsilon_0^{(n)} = \mathbf{x}_n, \quad n = 0, 1, 2, \cdots, \tag{5.10}$$

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \overline{\Delta\varepsilon_k^{(n)}}/\|\Delta\varepsilon_k^{(n)}\|^2, \qquad k, n = 0, 1, 2, \cdots. \tag{5.11}$$

These formulas are simpler than those for TEA in that they do not require an even-odd distinction, and VEA has also turned out to be the most useful member of this family of algorithms from the point of view of numerical computation. However, it suffers the (theoretical) defect that we do not know what system of linear equations is actually being solved by VEA. On the other hand, we do have an exactness result analogous to Theorems 2 and 3.

THEOREM 4 (McLeod, Graves-Morris). *Suppose a sequence* $\{\mathbf{x}_n\}$ *of vectors satisfies a recurrence relation of the form* (2.14) *for some coefficients* $c_0, c_1, \cdots, c_k$ ($c_0 \neq 0$, $c_k \neq 0$) *and some fixed vector* $\mathbf{s}$. *If the* VEA *array is generated from* $\{\mathbf{x}_n\}$ *by* (5.10) *and* (5.11), *and if all the required quantities exist, then*

$$\varepsilon_{2k}^{(n)} = \mathbf{s}, \quad n \geq 0, \quad \text{if } \sum_{j=0}^{k} c_j \neq 0, \tag{5.12}$$

and

$$\varepsilon_{2k}^{(n)}=\mathbf{0}, \quad n\geqq 0, \quad \text{if } \sum_{j=0}^{k} c_j=0. \tag{5.13}$$

Of course, we know from Theorem 1 that if the sequence is generated linearly, as in (2.1), then a recurrence of the form (2.14) exists, with k the degree of the minimal polynomial P, and with $c_k=1$. The conditon $c_0\neq 0$ is satisfied by either dropping zero terms and redefining k or starting the sequence with $\mathbf{x}_r$, where r is the multiplicity of 0 as a zero of P (see Lemma 1). Our assumption that 1 is not a root of P rules out the degenerate case (5.13).

Theorem 4 has an interesting history. The scalar case, essentially Theorem 2, was published over 20 years ago by Wynn [37], who conjectured the extension to the vector case. McLeod [25] proved the special case of Theorem 4 in which the coefficients c_j are restricted to be real numbers, even though the vector quantities might have complex entries. It may be difficult to imagine how an argument that proves certain relations among complex vectors with real coefficients might fail if the coefficients were allowed to be complex. In fact, McLeod's argument used techniques of universal algebra, first establishing a version of Theorem 4 for *matrix* quantities (with ordinary matrix inversion), then constructing a linear isomorphism of $\mathbb{C}^N$, as a *real* vector space, into a high-dimensional matrix space that would map Samelson inverses to matrix inverses. The construction depended on treating real and imaginary parts separately, and the mapping simply was not linear over $\mathbb{C}$.

That is where matters stood for more than a decade, in spite of the fact that everyone who knew about McLeod's theorem (including Wynn and McLeod) must have thought that the restriction to real scalars was merely an artifact of the indirect method of proof, and that a proper understanding of the theorem would remove it.

That understanding has recently been provided by Graves-Morris [19], who has related VEA to a general scheme for vector-valued rational interpolation and thereby obtained a constructive proof of Theorem 4 as a corollary. The details lie outside the scope of this paper.

6. Extension of the algorithms to nonlinear sequences. We turn our attention to a sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots$ generated by

$$\mathbf{x}_{j+1}=F(\mathbf{x}_j), \tag{6.1}$$

where F is a vector-valued function of real or complex vectors, defined on an open and connected domain D in N-space, and which has a Lipschitz continuous derivative. If $\mathbf{s}=F(\mathbf{s})$ is a fixed point in D, and $F'(\mathbf{s})$ is the Jacobian matrix of F at $\mathbf{s}$, then

$$F(\mathbf{x})-\mathbf{s}=F'(\mathbf{s})(\mathbf{x}-\mathbf{s})+O\left(\|\mathbf{x}-\mathbf{s}\|^2\right), \tag{6.2}$$

for all $\mathbf{x}\in D$. Note that this context includes that of the linearly generated sequences studied in the previous sections, as well as sequences generated by more general iterations of the form

$$\mathbf{x}_{j+1}=A\mathbf{x}_j+\mathbf{b}+\mathbf{e}_j, \tag{6.3}$$

where $\mathbf{e}_j$ is any small "error" that approaches $\mathbf{0}$ quadratically as $j\to\infty$. If $\mathbf{e}_j$ is the actual error in a linearly generated sequence (2.1), for example, from use of finite precision arithmetic, there is no reason to think that $\mathbf{e}_j\to\mathbf{0}$. Nevertheless, we will see

that the technique to be described here for nonlinear sequences has some practical application to linear sequences as well. That is, it has the effect of "squeezing out" small errors in the linear computations in many situations that are not covered by the theoretical development.

We assume as before that the Jacobian matrix $F'(\mathbf{s})$ does not have 1 as an eigenvalue, and we let k denote the degree of the minimal polynomial of $F'(\mathbf{s})$ with respect to $\mathbf{x}_0-\mathbf{s}$. All of the algorithms presented above may now be extended by "cycling" to generate a sequence $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2, \cdots$ of approximations to $\mathbf{s}$, in much the same way that scalar iterations are accelerated by iterating Aitken's Δ^2 method (Steffenson's method).

NONLINEAR EXTRAPOLATION ALGORITHM.

(a) Set $\mathbf{s}_0=\mathbf{x}_0$, the given starting vector, and $i=1$.
(b) Generate an appropriate number of vectors $\mathbf{x}$ by (6.1): $k+1$ for MPE or RRE, $2k$ for SEA, VEA, or TEA.
(c) Apply the appropriate algorithm to these iterates to compute a vector $\mathbf{s}^*$.
(d) Set $\mathbf{s}_i=\mathbf{s}^*$, increase i by 1, replace $\mathbf{x}_0$ by $\mathbf{s}^*$, and return to step (b).

Each time through this loop is called a *cycle*. (We want to avoid overuse of "iteration," which is already in use in reference to generation of the terms of the sequences, and which is explicitly involved in construction of any of the epsilon tables. Another type of iteration will be seen to be necessary for determination of k for MPE and RRE.) The principal "result" concerning this cycled extrapolation algorithm (all versions) is that it is quadratically convergent, in the following sense.

"THEOREM" 5. *Under the assumptions stated above on F and $\mathbf{s}$, if k is chosen on the ith cycle to be the degree of the minimal polynomial of $F'(\mathbf{s})$ with respect to $\mathbf{s}_{i-1}-\mathbf{s}$, and if $\mathbf{s}_0$ is sufficiently close to $\mathbf{s}$ in D, then*

$$\|\mathbf{s}_{i+1}-\mathbf{s}\|=O\left(\|\mathbf{s}_i-\mathbf{s}\|^2\right). \tag{6.4}$$

(In the case of the epsilon algorithms, we also assume that all the required quantities in the computations for each extrapolation step exist.)

Arguments for (6.4) in the VEA case were given independently by Brezinski [4] and Gekeler [17]. Skelboe [31] gave another argument in this case and also included the MPE and SEA cases. RRE may be treated similarly to MPE and TEA similarly to the other epsilon algorithms. The arguments are nearly persuasive, especially in the presence of strong empirical evidence for quadratic convergence, but unfortunately all of these authors have left a subtle gap, and we have not been able to close it, so we will instead *disclose* it. All of the arguments are based on linear propagation of errors through each extrapolation. In particular, quadratic deviations from the linear case are propagated as quadratic deviations from the exact extrapolations in the linear case. In the next two sections we will sketch Skelboe's arguments for the MPE and VEA cases, and then indicate the gaps. Since Skelboe's treatment of MPE is based on Cabay and Jackson's presentation of the algorithm [12] and ours is based on the simpler presentation in §2, our error analysis is also simpler, but the essential ideas remain the same.

7. Error analysis for MPE. Our immediate objective is to compare an extrapolated vector $\mathbf{s}^*$ from a sequence of the form (6.3) with the exact fixed point $\mathbf{s}$ extrapolated from the sequence (2.1), in both cases by application of MPE with $k=\deg P$, where P is the minimal polynomial of A with respect to $\mathbf{x}_0-\mathbf{s}$. To fix notation, we attach an

asterisk to all quantities that may contain an error

$$\mathbf{x}_{j+1}^* = A\mathbf{x}_j^* + \mathbf{b} + \mathbf{e}_j, \tag{7.1}$$

where $\mathbf{x}_0^* = \mathbf{x}_0 + \mathbf{e}_0$ is the given approximate value of the starting vector $\mathbf{x}_0$. We write

$$\mathbf{a}_j = \mathbf{x}_j^* - \mathbf{x}_j \tag{7.2}$$

for the errors in the data on which the extrapolation is based. Then $\mathbf{a}_0 = \mathbf{e}_0$, and

$$\mathbf{a}_{j+1} = A\mathbf{a}_j + \mathbf{e}_j = A^{j+1}\mathbf{e}_0 + \sum_{i=0}^{j} A^{j-i}\mathbf{e}_i. \tag{7.3}$$

Thus each of the **a**'s is a linear combination of the **e**'s and vice versa. We further write

$$\mathbf{u}_j^* = \Delta\mathbf{x}_j^* = \mathbf{u}_j + \Delta\mathbf{a}_j. \tag{7.4}$$

The extrapolation coefficients $\mathbf{c}^* = (c_0^*, c_1^*, \cdots, c_{k-1}^*)^t$ are computed as the least-squares solution of the overdetermined system

$$U^*\mathbf{c}^* = -\mathbf{u}_k^*, \tag{7.5}$$

where $U^* = [\mathbf{u}_0^*, \mathbf{u}_1^*, \cdots, \mathbf{u}_{k-1}^*]$. Thus, the residual vector $\mathbf{r}^*$, where

$$\mathbf{r}^* = \sum_{j=0}^{k} c_j^*\mathbf{u}_j^* \quad (\text{with } c_k^* = 1), \tag{7.6}$$

has minimal length, and $\mathbf{c}^*$ is given by

$$\mathbf{c}^* = -U^{*+}\mathbf{u}_k^*. \tag{7.7}$$

Finally, we have the extrapolated vector $\mathbf{s}^*$ given by

$$P^*(1)\mathbf{s}^* = \sum_{j=0}^{k} c_j^*\mathbf{x}_j^*, \tag{7.8}$$

where P^* is the "almost annihilating" polynomial

$$P^*(\lambda) = \sum_{j=0}^{k} c_j^*\lambda^j. \tag{7.9}$$

We write

$$\|\mathbf{a}\| = \max_{0 \le j \le k+1} \|\mathbf{a}_j\|. \tag{7.10}$$

Then we need to show that, for $\|\mathbf{a}\|$ sufficiently small, $P^*(1) \neq 0$ (so that $\mathbf{s}^*$ exists), and

$$\|\mathbf{s}^* - \mathbf{s}\| = O(\|\mathbf{a}\|). \tag{7.11}$$

The key to the argument is the perturbation analysis of least squares solutions, which may be found, for example, in [23, Chap. 9]. If we write $E = U^* - U$ and use the fact that $\operatorname{rank}(U) = k$, then [25, Thm. (9.12)] yields

$$\text{If } \|E\| \cdot \|U^+\| < 1, \text{ then } \operatorname{rank}(U^*) = \operatorname{rank}(U), \text{ and} \tag{7.12}$$
$$\|\mathbf{c}^* - \mathbf{c}\| \leqq \frac{\|U^+\|\left(\|E\|\|\mathbf{c}\| + \|\Delta\mathbf{a}_k\|\right)}{1 - \|E\| \cdot \|U^+\|}.$$

The matrix norms in this expression are given by the usual "spectral" norm

$$\begin{aligned}\|M\| &= \max\{\|M\mathbf{x}\| : \|\mathbf{x}\| = 1\} \\ &= \text{maximal singular value of } M.\end{aligned}$$

Since $\|E\| = O(\|\mathbf{a}\|)$, by (7.4), we have

$$\|\mathbf{c}^* - \mathbf{c}\| = O(\|\mathbf{a}\|). \tag{7.13}$$

It then follows that

$$|P^*(1) - P(1)| = \left|\sum_{j=0}^{k-1} \left(c_j^* - c_j\right)\right| \tag{7.14}$$
$$\leqq \sum_{j=0}^{k-1} |c_j^* - c_j| \leqq \sqrt{k}\,\|\mathbf{c}^* - \mathbf{c}\| = O(\|\mathbf{a}\|).$$

In particular, for $\|\mathbf{a}\|$ sufficiently small, $P^*(1) \neq 0$, since $P(1) \neq 0$. Finally, we combine (7.13) and (7.14) to compute

$$\begin{aligned}|P^*(1)| \cdot \|\mathbf{s}^* - \mathbf{s}\| &= \|P^*(1)\mathbf{s}^* - P^*(1)\mathbf{s}\| \\ &\leqq \|P^*(1)\mathbf{s}^* - P(1)\mathbf{s}\| + |P^*(1) - P(1)| \cdot \|\mathbf{s}\| \\ &= \left\|\sum_{j=0}^{k} \left(c_j^*\mathbf{x}_j^* - c_j\mathbf{x}_j\right)\right\| + O(\|\mathbf{a}\|) \\ &= \left\|\sum_{j=0}^{k} \left[\left(c_j^* - c_j\right)\mathbf{x}_j^* + c_j\left(\mathbf{x}_j^* - \mathbf{x}_j\right)\right]\right\| + O(\|\mathbf{a}\|) \\ &\leqq \sum_{j=0}^{k} |c_j^* - c_j|\,\|\mathbf{x}_j^*\| + \sum_{j=0}^{k} |c_j| \cdot \|\mathbf{a}_j\| + O(\|\mathbf{a}\|) \\ &= O(\|\mathbf{a}\|).\end{aligned}$$

Since $P^*(1) \neq 0$, we also have

$$\|\mathbf{s}^* - \mathbf{s}\| = O(\|\mathbf{a}\|), \tag{7.15}$$

as desired.

Now we turn to the "proof" of "Theorem" 5 in the MPE case. Given a possibly nonlinear sequence (6.1) satisfying (6.2), we may consider the sequence to be of the form (6.3) with $A = F'(\mathbf{s})$, i.e. of linear type with errors that are $O(\|\mathbf{x}_j - \mathbf{s}\|^2)$, thus also $O(\|\mathbf{x}_0 - \mathbf{s}\|^2)$. Since these errors are propagated linearly in the sense of (7.11), the error

in the extrapolated value at the end of each cycle is $O(\|\mathbf{x}_0-\mathbf{s}\|^2)$, where $\mathbf{x}_0$ is the starting vector for that cycle.

The problem with this argument is that the key estimate for the extrapolation, (7.13), has an implied proportionality constant, and (7.12) reveals that the constant depends on $\|U^+\|$, which is constant for a given cycle, but certainly not from one cycle to the next. In fact, the difference matrices U tend to become more and more singular as $\mathbf{s}$ is approached, and there is no indication that $\|U^+\|$ will remain bounded, or that $\|E\|\cdot\|U^+\|$ will be bounded away from 1. When these additional conditions are met, we do see quadratic convergence of the $\{\mathbf{s}_i\}$ in practice, but this is not always the case.

8. Error analysis for VEA. Here we show that errors in the epsilon table defined by (5.10)–(5.11) vary linearly with errors in the initial data (5.10). As in the previous section, we use asterisks to indicate values actually computed

$$\boldsymbol{\varepsilon}_k^{(n)*}=\boldsymbol{\varepsilon}_k^{(n)}+\mathbf{a}_k^{(n)}, \tag{8.1}$$

where $\boldsymbol{\varepsilon}_k^{(n)}$ is the exact (k,n)-entry, and $\mathbf{a}_k^{(n)}$ is the error in that entry. Then the errors in the initial data are

$$\mathbf{a}_0^{(n)}=\mathbf{x}_n^*-\mathbf{x}_n, \tag{8.2}$$

and, as in §7, we write

$$\|\mathbf{a}\|=\max_{0\le n\le 2K}\|\mathbf{a}_0^{(n)}\|, \tag{8.3}$$

where K is the appropriate value of k to achieve the exactness result of Theorem 4. We show by induction that

$$\|\mathbf{a}_k^{(n)}\|=O(\|\mathbf{a}\|) \tag{8.4}$$

for all k, n, under the assumption that all the required entries of the epsilon table exist. In particular, this will be the case for $\mathbf{a}_{2K}^{(0)}$, the error in the extrapolated value for a single cycle.

The computed entries of the epsilon table satisfy

$$\boldsymbol{\varepsilon}_{k+1}^{(n)*}=\boldsymbol{\varepsilon}_{k-1}^{(n+1)*}+\Delta\bar{\boldsymbol{\varepsilon}}_k^{(n)*}/\|\Delta\boldsymbol{\varepsilon}_k^{(n)*}\|^2. \tag{8.5}$$

If we subtract the exact values (5.11), we get a recursion for the errors

$$\begin{aligned}\mathbf{a}_{k+1}^{(n)}&=\mathbf{a}_{k-1}^{(n+1)}+\frac{\Delta\bar{\boldsymbol{\varepsilon}}_k^{(n)*}}{\|\Delta\boldsymbol{\varepsilon}_k^{(n)*}\|^2}-\frac{\Delta\bar{\boldsymbol{\varepsilon}}_k^{(n)}}{\|\Delta\boldsymbol{\varepsilon}_k^{(n)}\|^2}\\&=\mathbf{a}_{k-1}^{(n+1)}+\left[(\alpha-1)\Delta\bar{\boldsymbol{\varepsilon}}_k^{(n)}+\alpha\Delta\bar{\mathbf{a}}_k^{(n)}\right]/\|\Delta\boldsymbol{\varepsilon}_k^{(n)}\|^2,\end{aligned} \tag{8.6}$$

where

$$\alpha=\|\Delta\boldsymbol{\varepsilon}_k^{(n)}\|^2/\|\Delta\boldsymbol{\varepsilon}_k^{(n)*}\|^2. \tag{8.7}$$

For $\|\mathbf{a}\|$ sufficiently small, the induction hypothesis and the assumption that $\|\Delta\boldsymbol{\varepsilon}_k^{(n)}\| \neq 0$ (for all k and n) imply that $\|\Delta\boldsymbol{\varepsilon}_k^{(n)*}\|$ may be bounded away from 0 (for all k and n). Thus α and all the entries of the *computed* ε table exist.

$$\alpha - 1 = \left(\|\Delta\boldsymbol{\varepsilon}_k^{(n)}\|^2 - \|\Delta\boldsymbol{\varepsilon}_k^{(n)*}\|^2\right) / \|\Delta\boldsymbol{\varepsilon}_k^{(n)*}\|^2. \tag{8.8}$$

Substitution from (8.1) and expansion of the squared norms in the numerator of (8.8) by inner products lead to the observation that

$$\alpha - 1 = O\left(\|\mathbf{a}_k^{(n)}\|\right) = O(\|\mathbf{a}\|),$$

so (8.6) completes the induction step to show that (8.4) is valid.

The argument that takes us from (8.4) to "Theorem" 5 is exactly the same as that given in the previous section for the MPE case. The gap to be filled is more subtle in this case, however. Formula (6.4) requires an absolute constant, not one dependent on the cycle number, and that has not yet been demonstrated. For example, the degree of the minimal polynomial of $F'(\mathbf{s})$ with respect to $\mathbf{x}_0 - \mathbf{s}$ depends on $\mathbf{x}_0$, and that affects the assumption of existence of entries in the epsilon array. The possible degrees cannot exceed the dimension of the space, of course, and for any finite number of cycles, there are only a finite number of instances of (8.4) to consider. But (6.4) is a statement about *asymptotic* behavior, and, as is the case with MPE, the epsilon computations tend to become more singular (both numerically and in theory) as the limit is approached. What "Theorem" 5 actually provides is a strong indication, supported by much numerical evidence, that the sequences $\{\mathbf{s}_i\}$ converge rapidly from a good starting point, usually to the limits of machine accuracy before any singularity problems set in.

9. Strategies for implementation. In this section we will consider our "base sequence" $\{\mathbf{x}_j\}$ to be generated by an iteration of the form (6.3), which we may think of as incorporating both the nonlinear case (6.1), with $A = F'(\mathbf{s})$ and subject to (6.2), and the linear case (2.1) with small errors, for example, from limited precision of data and/or computations. We discuss here some considerations for practical use of the various extrapolation techniques.

The first such consideration has already been noted in §2: If little is known about the starting point, it may be useful to generate some number m of base iterates that will *not* be used in the extrapolation, effectively starting over with $\mathbf{x}_m$ in place of $\mathbf{x}_0$. This number may be small or large, depending on the cost of computing individual terms. One purpose of this initial step is to reduce the (theoretical) size k of the extrapolation problem by removing factors of λ from the minimal polynomial (if there are any). Of course, in the process of eliminating the role of 0 as an eigenvalue, one is also reducing the influence of eigenvalues close to 0. (Think of the base sequence as having the form (4.13), at least approximately.) Examination of these initial iterates may provide some evidence of the rate of convergence (or divergence!). If the base sequence appears to converge rapidly, no extrapolation may be needed. On the other hand, if the sequence appears to diverge, the selected sequence transformation should probably be applied starting from $\mathbf{x}_0$, to avoid numerical difficulties with data ranging over many orders of magnitude.

Up to now we have said nothing about determination of the "magic number" k (degree of the minimal polynomial), and in fact there is no practical way to determine k in advance. Fortunately, it is not necessary to do so. Cabay and Jackson [12] have observed that even poor approximations to k and to the coefficients $\mathbf{c}$ of P can lead to

good approximations (via MPE) to $\mathbf{s}$, and the same is true of the other extrapolation methods. The reason for this is evident from the eigenvector decomposition (4.13). For $k < \deg P$, only the "dominant" components appear in (4.13), and a small error term is neglected; instead of achieving equality in (2.11), the least squares solution (2.12) gives coefficients of an "almost annihilating" polynomial that is the "best" monic polynomial of degree k for eliminating the influence of k dominant components of the error. (Similar comments apply to RRE and the epsilon algorithms, but the sense of "best" depends on the method. Further details of this interpretation of the methods will be provided in the next section.)

Thus an appropriate way to implement MPE and RRE on the first cycle is to extrapolate to $\mathbf{s}_{m,k}$ from $\mathbf{x}_m, \mathbf{x}_{m+1}, \cdots, \mathbf{x}_{m+k+1}$ with $k = 1, 2, 3, \cdots$, and stop when the least squares residuals, or the solution residuals $\mathbf{s}_{m,k} - F(\mathbf{s}_{m,k})$, are acceptably small. When there is strong separation between the "dominant" and the "small" eigenvalues, there is often a precipitous drop in the magnitudes of these residuals when k reaches the number of dominant ones (multiplicities included). In other cases the decline in residuals with increasing k is gradual. However, with cycling, even a k large enough to produce only one or two orders of magnitude difference between $\|\mathbf{s}_{m,k}\|$ and $\|\mathbf{s}_{m,k} - F(\mathbf{s}_{m,k})\|$ will be sufficient to produce convergence of the extrapolation sequence. If k is too small, we cannot expect this convergence to be quadratic, but it may still be far faster than the convergence of the base sequence (if any).

There is no absolute guarantee that the k accepted on the first cycle will continue to work on subsequent cycles, but in practice it always seems to. If k is actually $\deg P$ on the first cycle, then with high probability it accounts for the error components corresponding to all the eigenvalues of A, since there are only a finite number of special directions for $\mathbf{x}_0 - \mathbf{s}$ in which any could be missed. In this case, the required degree could not increase on subsequent cycles. In practice, the number of dominant eigenvalues whose error contribution must be suppressed remains constant from one cycle to the next. Thus for MPE and RRE multiple extrapolations with increasing k are unnecessary after the first cycle.

The epsilon algorithms, on the other hand, provide much less definite evidence of the proper degree k. In the linear case, Theorems 2, 3, and 4 produce (in theory) an entire column (the $2k$th) of equal results for SEA, TEA, and VEA, respectively, so computing the appropriate inverses for the next column should be impossible. However, this does not always happen, as a result of roundoff error. Furthermore, individual inverses can fail to exist for other reasons, especially with SEA, which requires a significant difference in *every* component for adjacent entries in each column. Even if a suitable $k < N$ is found (for example, by acceptably small residuals in the $2k$th column), this does not shorten the computation on subsequent cycles, because of the recursive nature of the epsilon formulas (in contrast to MPE and RRE, which are explicit once k is known).

It is easily seen that the computational cost of a single epsilon table out to column $2k$ is comparable (for large k and N) to the cost of a single-extrapolation cycle for MPE or RRE with an N by k least squares step. Thus the epsilon extrapolations have a computational advantage (given the base sequence) on the first cycle, but not on subsequent cycles. Offsetting that modest advantage, the epsilon algorithms require nearly twice as many base iterates. We have found, for example, in applications of these methods to the solution of partial differential equations with finite difference grids of even moderate size, that 80 to 90 percent of the total computation time may be devoted to computation of the base iterates.

Just as there may be an advantage in generating some number of base iterates at the start that will not be used in the extrapolation, there may also be a good reason for inter-cycle "runs" of the base sequence generator (if it is convergent). That is, instead of using the extrapolated vector $\mathbf{s}^*$ as the new $\mathbf{x}_0$ for the next cycle, it could be used as the start of a sequence of base iterates $\mathbf{x}_1, \mathbf{x}_2, \cdots$, with $\mathbf{x}_m$ taken as the start of the next extrapolation cycle. (The choice of m could vary with the cycle as well.) One reason for this strategy is that an extrapolation may magnify the error components associated with small eigenvalues at the same time it is drastically reducing the dominant components. The base iteration reduces the small components rapidly, while not increasing the dominant ones.

Another modification of the method that has been found to be effective in some circumstances is to use every qth base iterate for some $q>1$, i.e. to extrapolate from $\mathbf{x}_0, \mathbf{x}_q, \mathbf{x}_{2q}, \cdots$. This is equivalent to replacing A with A^q, thus achieving greater separation of the dominant from the other eigenvalues.

10. Connections and credits. We have commented briefly on the work of some individuals in connection with the ideas presented. Here we attempt to fill in more of the details of who did what and how these efforts relate to each other.

Cabay and Jackson [12] introduced the minimal polynomial extrapolation (so named by Skelboe [31]), but in the form of (2.16), which is a little more complicated to compute than our MPE. They trace the underlying idea to Schmidt [28] and note that little had previously come of Schmidt's ideas except the direction taken by Shanks [29] and Wynn [35] with the e-transform and epsilon algorithm. Independently of Cabay and Jackson, Mešina [26] described an acceleration algorithm that was essentially MPE, except he chose a polynomial whose coefficients sum to 1 rather than a monic polynomial. Both [12] and [26] dealt with the case of a linearly generated base sequence only; Mešina used cycling and inter-cycle runs of the sequence generator, whereas Cabay and Jackson used neither. Mešina used least squares solutions of the inconsistent equations; Cabay and Jackson preferred the Krylov method [34, pp. 369–377] for its computational simplicity while still achieving "nearly minimal" residuals, as opposed to the actual minimization of least squares.

Skelboe [31] was the first (in print) to bring together the work of Brezinski and Rieu [11] and Gekeler [17] on application of the (cycled) vector epsilon algorithm to nonlinearly generated sequences with a cycled version of MPE (based on [12], not [26]) for the same purpose.

Many authors have addressed the question of the appropriate extension of Aitken's Δ^2 to vector sequences. Henrici [22, pp. 115–117] presents a cycled version of the full rank extrapolation (3.1) as "Steffenson's iteration for systems of equations" and cites empirical evidence for quadratic convergence. (*Steffenson's iteration* in the scalar case is the cycled version of Aitken's Δ^2.) Ortega and Rheinboldt [27, pp. 199–200] note that the Henrici formulation is equivalent to one given by Ludwig [24]. Eddy has unpublished work on the full rank extrapolation, also dating to the early 1950's, which eventually led to his work [13], [14] on the reduced rank extrapolation (RRE), and it is his presentation we followed in §§2 and 3 for both MPE and RRE. We can now demonstrate how closely related these two methods actually are, and in the process add another insight into how they work.

We return to equations (3.2) and (3.3) *without* the assumption that k is the degree of a minimal polynomial, and we seek an approximation $\mathbf{s}^*$ to $\mathbf{s}$ such that

$$\mathbf{s}^* = \mathbf{x}_0 + \sum_{j=0}^{k-1} \xi_j \mathbf{u}_j, \tag{10.1}$$

where the vector $\xi=(\xi_0,\xi_1,\cdots,\xi_{k-1})^t$ is chosen to minimize the norm

$$\|\mathbf{u}_0+V\xi\|, \tag{10.2}$$

i.e. the length of the residual vector in (3.3). It is easy to see that (10.1) may be rewritten as

$$\mathbf{s}^*=\sum_{j=0}^{k}\gamma_j\mathbf{x}_j, \tag{10.3}$$

with

$$\begin{aligned}&\gamma_0=1-\xi_0,\\ &\gamma_j=\xi_{j-1}-\xi_j \quad \text{for } 1\leqq j\leqq k-1,\\ &\gamma_k=\xi_{k-1},\end{aligned}$$

and thus

$$\sum_{j=0}^{k}\gamma_j=1. \tag{10.4}$$

Furthermore, the condition of minimizing (10.2) uniquely determines ξ and therefore the set of coefficients $\gamma_0,\cdots,\gamma_k$ in (10.3) satisfying (10.4).

Now

$$\mathbf{u}_0+V\xi=\mathbf{u}_0+\sum_{j=0}^{k-1}\xi_j\mathbf{v}_j=\Delta\left(\mathbf{x}_0+\sum_{j=0}^{k-1}\xi_j\mathbf{u}_j\right)$$

$$=\Delta\left(\sum_{j=0}^{k}\gamma_j\mathbf{x}_j\right)=\sum_{j=0}^{k}\gamma_j\mathbf{u}_j.$$

Thus RRE (for arbitrary k) is equivalent to choosing an approximation $\mathbf{s}^*$ of the form (10.3) by solving the problem

$$\text{Minimize}\left\|\sum_{j=0}^{k}\gamma_j\mathbf{u}_j\right\| \quad \text{subject to } \sum_{j=0}^{k}\gamma_j=1. \tag{10.5}$$

But this is precisely Mešina's algorithm. Thus the essential distinction between MPE and RRE in determining coefficients by minimization is whether one chooses to set the leading coefficient equal to 1 or the sum of the coefficients equal to 1. It is not surprising, then, to find that these two algorithms behave very much alike in most circumstances, but that the Eddy-Mešina algorithm is sometimes less stable numerically, since it tends to produce very large coefficients with alternating signs. (We explained this instability earlier, in the context of the Eddy formulation, by the use of second differences for the coefficients in the least squares problem.)

Since both MPE and RRE lead to extrapolations of the form (10.3)–(10.4), with $\gamma_j = c_j/P(1)$ in the MPE case, it is natural to ask how we might find the *best* such extrapolation for a given k, in the sense of minimizing $\|\mathbf{s}-\mathbf{s}^*\|$. For any linear combination of the form (10.3), using (2.2) and (10.4), we have

$$\begin{aligned}
\mathbf{s}-\mathbf{s}^* &= (I-A)^{-1}\mathbf{b} - \sum_{j=0}^{k} \gamma_j \mathbf{x}_j \\
&= (I-A)^{-1}\left[\mathbf{b} - \sum_{j=0}^{k} \gamma_j (I-A)\mathbf{x}_j\right] \\
&= (I-A)^{-1}\left[\mathbf{b} - \sum_{j=0}^{k} \gamma_j (\mathbf{x}_j - \mathbf{x}_{j+1} + \mathbf{b})\right] \\
&= (I-A)^{-1}\left[\mathbf{b} + \sum_{j=0}^{k} \gamma_j \mathbf{u}_j - \left(\sum_{j=0}^{k} \gamma_j\right)\mathbf{b}\right] \\
&= (I-A)^{-1} \sum_{j=0}^{k} \gamma_j \mathbf{u}_j .
\end{aligned}$$

Thus

$$\|\mathbf{s}-\mathbf{s}^*\| \leqq \|(I-A)^{-1}\| \left\| \sum_{j=0}^{k} \gamma_j \mathbf{u}_j \right\| . \tag{10.6}$$

Now the first factor on the right is unknown but constant (only approximately so in the nonlinear case). Thus, to come as close as possible to a minimum for the left-hand side, it is natural to minimize the second factor on the right. This is in the spirit of Mešina's derivation, which attempts to minimize the norm of the residual,

$$\|A\mathbf{s}^* + \mathbf{b} - \mathbf{s}^*\| .$$

Essentially the same calculation shows that this quantity is equal to the second factor on the right in (10.6).

All vector norms in the preceding discussion have been Euclidean norms, but as observed in a recent work [30] by the authors, any other norm would do as well, as long as the minimization problem could be solved efficiently. In particular, if either the l_1 norm or the l_∞ norm were used, the least squares problem would be replaced by a linear programming problem. Both of these norms have been proposed in [30] as alternatives to the Euclidean norm.

There is an extensive literature on polynomial extrapolation methods, some of which are much more familiar than MPE and RRE, for example, the conjugate gradient and Chebyshev methods. Since notation and terminology differ markedly from one author to the next, we will place the methods studied here in the context of the broader range of polynomial methods.

Given a linear base sequence generator (2.1) with a unique fixed point $\mathbf{s}$ given by (2.2), suppose $\mathbf{s}^*$ is *any* "extrapolation" formed from a weighted average of $\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k$ for any k (temporarily fixed)

$$\mathbf{s}^* = \sum_{j=0}^{k} \gamma_j \mathbf{x}_j, \qquad \sum_{j=0}^{k} \gamma_j = 1. \tag{10.7}$$

As we saw in the proof of Theorem 1,

$$\mathbf{x}_j - \mathbf{s} = A^j(\mathbf{x}_0 - \mathbf{s}), \qquad j = 0, 1, 2, \cdots,$$

so

$$\mathbf{s}^* - \mathbf{s} = \sum_{j=0}^{k} \gamma_j(\mathbf{x}_j - \mathbf{s}) = \sum_{j=0}^{k} \gamma_j A^j(\mathbf{x}_0 - \mathbf{s}) = P(A)(\mathbf{x}_0 - \mathbf{s}),$$

where P is the polynomial defined by

$$P(\lambda) = \sum_{j=0}^{k} \gamma_j \lambda^j. \tag{10.8}$$

That is, the error in the extrapolation is obtained from the initial error by applying a polynomial in A, where A is the iteration matrix and the coefficients of the polynomial are the averaging weights. (The notation in (10.8) differs from the minimal polynomial notation in §2 by a normalizing factor of $P(1)$. It is more convenient here to have coefficients as in (10.7) than to have a monic polynomial.) Hageman and Young [21, pp. 40–41] call any method satisfying this error condition a *polynomial acceleration method*. Of course, one would like to choose P to make

$$\|\mathbf{s}^* - \mathbf{s}\| = \|P(A)(\mathbf{x}_0 - \mathbf{s})\| \tag{10.9}$$

as small as possible. For the "right" degree k, MPE and RRE make this error exactly zero, and for smaller k (depending on the number of "dominant" eigenvalues of A) they tend to make it small enough to qualify as legitimate extrapolation methods.

It follows from (10.9) that

$$\|\mathbf{s}^* - \mathbf{s}\| \leqq \|P(A)\| \|\mathbf{x}_0 - \mathbf{s}\|, \tag{10.10}$$

for the induced matrix norm, and some polynomial methods are based on minimizing $\|P(A)\|$ subject to constraints on effective computability of the coefficients (weights). One cannot assure that $\|P(A)\| = 0$ unless k is the degree of the characteristic polynomial of A, which will often be the dimension N of the vector space. The vector norm in (10.10) need not be Euclidean, of course. Hageman and Young have given an excellent (unified) treatment of the Chebyshev and conjugate gradient (CG) methods [21, Chaps. 4 and 7, respectively] in the case of a *symmetrizable* iteration matrix A, that is, where $W(I - A)W^{-1}$ is symmetric and positive definite for some invertible matrix W. The choice of appropriate vector norm depends on W. In the Chebyshev case what

is actually minimized is the "virtual spectral radius" of $P(A)$

$$\max_{m(A)\leq\lambda\leq M(A)} |P(\lambda)|,$$

where $m(A)$ and $M(A)$ are the smallest and largest eigenvalues, respectively. In the CG case, the minimized quantity, expressed in Euclidean norm, is

$$\left\| [W^t W(I-A)]^{1/2}(\mathbf{s}^*-\mathbf{s}) \right\|.$$

In both cases the polynomials P_k for increasing degree k satisfy a three-term recurrence from which the successive extrapolants $\mathbf{s}_k^*$ may be computed recursively, without explicit determination of the coefficients $\gamma_{i,k}$. However, these methods require restrictions on the matrix A and some knowledge of its eigenvalues for effective use, and this is not the case for MPE or RRE. ([21, Chap. 12] summarizes what is known about generalizations of the Chebyshev and CG methods to nonsymmetrizable iterations.)

The explicit solution in the proof of Theorem 1 of the recursion (2.1) may also be written

$$\mathbf{x}_j = A^j\mathbf{x}_0 + (I-A)^{-1}(I-A^j)\mathbf{b},$$

from which we see that $\mathbf{s}^*$ of the form (10.7) may be written

$$\mathbf{s}^* = P(A)\mathbf{x}_0 + R(A)\mathbf{b}, \tag{10.11}$$

where P is given by (10.8) and $R(\lambda)=[1-P(\lambda)]/(1-\lambda)$, also a polynomial. (Again, this differs from R as defined in §2 by the normalizing factor $P(1)$.)

Some authors have taken (10.11) as the definition for polynomial extrapolation methods (see, for example, [18]). Faddeev and Faddeeva [15] use the term "universal algorithm" for such methods when the coefficients are determined independently of the particular sequence being accelerated. (We would call such a method "linear," in contrast to the methods studied here, for which each $\mathbf{s}^*$ is formed in a nonlinear way from the given vector sequence.) They show that, with strong assumptions on the locations of eigenvalues (confined to an interval on the real line), both maximal suppression of components of the error vector and minimization of residuals lead to Chebyshev-type methods (the classical method in the first case). By focusing on the spectral radius rather than the norm of $P(A)$, Germain-Bonne [18] has shown that methods not depending on symmetry or real eigenvalues may be devised, but at the expense of needing to know approximate locations in the complex plane of all the eigenvalues.

Vorobyev [33] makes a distinction between methods of the form (10.11), presumably in the "universal" sense, and methods for which successive extrapolations $\mathbf{s}_m^*$ are computed by

$$\mathbf{s}_m^* = \mathbf{x}_m + \sum_{j=0}^{k-1} \beta_j \mathbf{u}_{m+j}, \tag{10.12}$$

where k is fixed (to suppress the error components corresponding to the k largest eigenvalues) and the coefficients are to be determined directly from the sequence of vectors. Note that the Cabay–Jackson form of MPE and RRE, as given by (2.16) and (3.4), respectively, both have this form when the starting point is allowed to "slide" along the base sequence, and the $(2k)$th column of each of the epsilon arrays implicitly

has this form as well. Vorobyev also writes the problem in a form equivalent to that of Theorem 1 (i.e., MPE), with coefficients to be determined from a system of equations equivalent to (2.11) "in one way or another." The idea is not pursued further, and no reference is given, but he calls this "the *extrapolation* method." Instead, Vorobyev determines the coefficients for his accelerated sequence by the "method of moments," which involves successive projections on certain subspaces and orthogonalizations of these projections. He shows that, for a self-adjoint A, the error $\|\mathbf{s}-\mathbf{s}_m^*\|$ is $O(|\lambda_{k+1}|^m)$, where λ_{k+1} is the $(k+1)$th largest eigenvalue.

All of the methods studied here have "sliding" (as opposed to "cycling") versions, applicable primarily to linearly generated sequences of the form (2.1). Elsewhere [30] we introduce a sliding method that is conceptually and computationally simpler than either MPE or the method of moments and that has a similar error analysis.

11. Numerical examples. We have conducted numerical tests of the five methods described above with many linear and nonlinear vector iteration problems drawn from a variety of sources: [1], [3]–[5], [7], [11], [12], [16], [17], [20], [36]. Cabay and Jackson [12] have reported extensive numerical comparisons of their version of MPE with SEA and VEA in dimensions 10 and 400, but limited to linear iterations and without cycling. Mešina [26] reported having used his equivalent of RRE for a problem of dimension about 3000 in neutron transport theory and claimed an average reduction in the number of base iterates of 3 to 5 times. He also presented a number of comparisons of his method (in dimension 50) using cycling and inter-cycle runs, with the Chebyshev method, also limited to linear base iterations. Eddy has informed us in a private communication that he has applied RRE to "the vector sequences generated by various finite difference equivalents of a Poisson equation (10×10 mesh in 2D, $10\times10\times10$ mesh in 3D)$\cdots$ [and] cut computation time by a factor ranging from 2.4 to 4.7 as compared to using no extrapolation at all." Skelboe's numerical examples [31] arise from steady state analysis of electrical circuits. The iterations are nonlinear and of low dimension, with an appropriate degree k predictable from the circuit. He includes MPE, SEA and VEA in favorable comparisons with Newton and gradient methods previously used for such circuit analysis; in most cases SEA is effective and the most efficient.

We present here selected examples from our numerical comparisons which serve to illustrate several general, but tentative, conclusions. We have found no effective way to implement TEA, either by iteration (5.4)–(5.6) or by solving the defining equations (4.2) and (4.4) directly. However the auxiliary vector $\mathbf{y}$ is chosen (including "at random"), we seldom find any significant acceleration of convergence. When applied to divergent sequences, it tends to slow divergence somewhat, but not to produce a convergent sequence. We are not aware of any other authors reporting success with TEA, either, so it remains for us a theoretical bridge between the MPE-type algorithms (via approximate solution of inconsistent linear equations) and the epsilon-type algorithms.

The anticipated numerical difficulties with RRE arise frequently, and we have found very few examples of performance that is noticeably better than MPE. Similarly, the numerical difficulties with SEA are real (and documented elsewhere [11]), and this method seldom performs better than VEA. Under most circumstances, MPE has an advantage over VEA in requiring only half as many base iterates and less storage (for a given accuracy), at the cost of more computation on the first cycle (to find k). However, this advantage is sometimes offset by numerical considerations, as we shall see.

In all of the figures presented here, we show the minimal number of significant digits of accuracy (SD) in each extrapolated answer as a function of the number of base iterations (2.1) or (6.1). SD is computed from the formula

$$\text{SD} = -\log_{10}(\|X - \text{ANS}\|_\infty / \|\text{ANS}\|_\infty), \tag{11.1}$$

where X is the computed extrapolation and ANS is the known answer. Inter-cycle steps are indicated by solid lines, and steps within cycles (increasing k), if shown at all, are indicated by dashed lines.

Example 1. *Divergent iteration with an eigenvalue close to unity.* Wynn [36] illustrates SEA and VEA with sequences derived from a number of well known iterative methods. In particular, we consider the solution of the system

$$\begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 23 \\ 32 \\ 33 \\ 31 \end{bmatrix}$$

by diagonal relaxation [36, Eq. (12)] starting from $\mathbf{x}_0 = (0,0,0,0)^t$. The problem is the same as [36, Eq. (16)], except for correction of a typographical error, and the solution is $\mathbf{s} = (1,1,1,1)^t$. When written in the form (2.1), the matrix A generating the linear iteration has eigenvalues (approximately) -2.48, 0.9985, 0.915, 0.562. The one large root causes the base iteration to be divergent from any starting point, and the root near 1 causes $I - A$ to be nearly singular, a situation that Cabay and Jackson claim leads to superior performance for MPE.

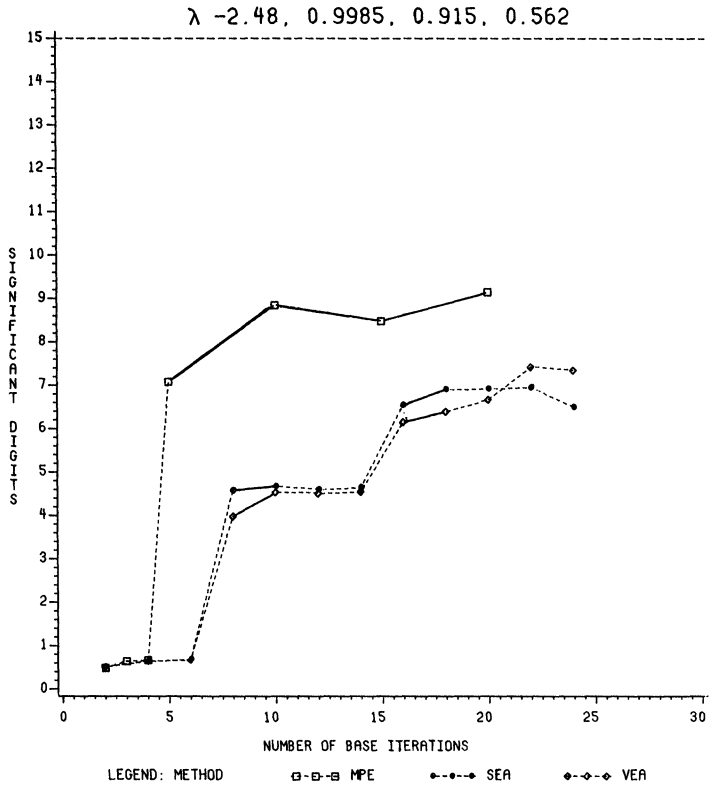

FIG. 1. *Divergent* 4 *by* 4.

Figure 1 illustrates the following points: (a) For the methods shown (MPE, VEA, SEA), the jump in accuracy from $k=3$ to $k=4$ is striking; (b) VEA and SEA are very similar; (c) in spite of the linearity of the base iteration, a significant gain in accuracy can be achieved by a second cycle with all three methods; (d) there is no gain after the second cycle; (e) MPE is more accurate, and achieves its accuracy faster, than the epsilon methods, in agreement with Cabay and Jackson (but see Example 3 for another view of near-singularity).

RRE is not shown in Fig. 1 because it fails to converge to the right answer, and therein lies a tale. We first computed this and many other examples using APL code on an IBM 5100. With apparently equivalent FORTRAN code and essentially the same (double) precision, we expected to get the same answers. In most cases we did, but there were several cases like this one, in which RRE or SEA or both failed to give the performance in FORTRAN that we had found in APL, apparently because of the greater sensitivity of these two methods to the internal handling of numbers and to the differences between the APL "domino" operator and IMSL's least squares routine. The APL answers for RRE were nearly identical to those for MPE.

The steps for $k<4$ are shown only for the first cycle for MPE, since these would not be computed on subsequent cycles. On the other hand, these steps are routinely available on all cycles for the epsilon algorithms, because of the recursive nature of the calculations, so they have been shown.

Example 2. *More rapid divergence.* Wynn also uses the Gauss–Seidel iteration as an example. When this method is applied to [36, Eq. (14)], and the iteration is written in the form (2.1), we find eigenvalues $-2.35 \pm 2.05i$, -0.0228, and 0. The results are

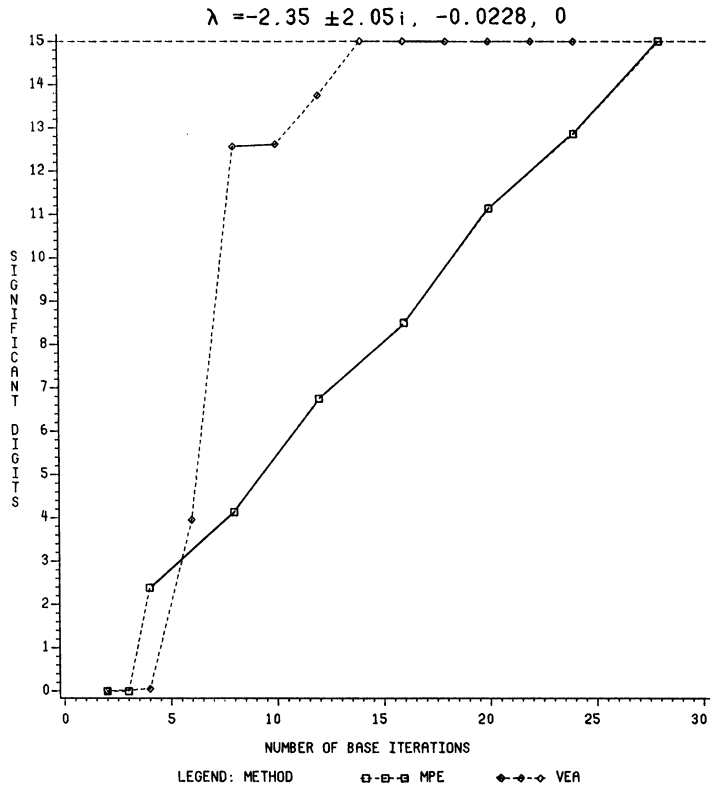

FIG. 2. *Divergent* 4 *by* 4.

shown in Fig. 2. Here the base iteration diverges more rapidly than in Example 1, which seems to affect MPE much more than VEA. RRE and SEA fail to detect the unique solution to this problem by "converging" quickly to a vector on which the sequence generator is almost constant, approximately

$$(13.36, -1.940, 5.532, -5.342)^t.$$

The sequence eventually diverges from this point, of course, but the first six iterates agree to 14 SD, so the extrapolation is to the same point.

The MPE cycles in Fig. 2 have only three steps instead of four because the method correctly detects $k=3$; the system of equations becomes singular with $k=4$. VEA does not detect this, but nevertheless achieves an excellent result on the first cycle and machine accuracy on the second. MPE also achieves machine accuracy eventually, but the convergence is only linear, even when the cycle starting point is quite close to the right answer.

Example 3. *Convergent iterations with eigenvalues close to unity.* In this example and the next two we use Cabay and Jackson's 10 by 10 problem generator [12, §§6, 7]. The eigenvalues occur in complex conjugate pairs $c_i e^{\pm\theta_i}$, where $\theta_i=\pi(i-1)/4$, and $i=1, 2, 3, 4, 5$. Thus the real roots ($i=1$ and 5) are double roots, and with all $c_i\neq 0$, the degree k is 8. For this example, we take $c_2=0.4$, $c_3=0.2$, $c_4=0.1$, and $c_5=0.001$. Figure 3 shows results for MPE and VEA, both with $k=8$, for five choices of c_1 approaching 1.0 (top to bottom in Fig. 3). Observe that in all cases VEA achieves a superior result on the first cycle and that the two methods are very similar on second

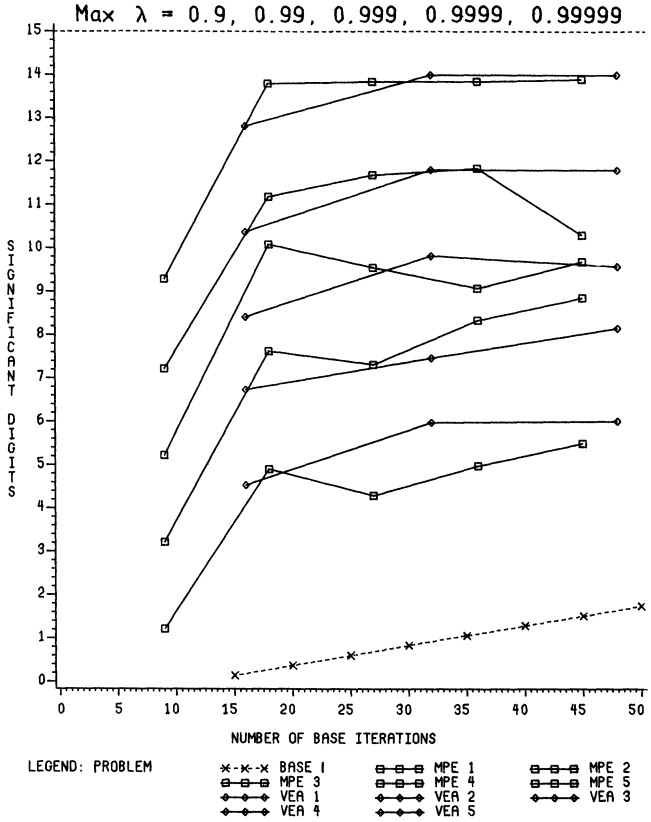

FIG. 3. *Five convergent* 10 *by* 10*s*.

and subsequent cycles. If anything, VEA has a slight edge in eventual accuracy in the *worst* case, in contrast to Cabay and Jackson's observation about superior performance of MPE in the near-singular case. Of course, the efficiency of MPE in use of base iterates is still evident here.

Example 4. *Varying the degree k.* Here we use the same problem generator as in Example 3, with $c_1 = 0.4$, $c_2 = 0.2$, $c_3 = 1.0$, $c_4 = 0.1$, $c_5 = 0.01$. Thus the eigenvalues of largest magnitude are $\pm i$. (Only $\lambda = 1$ is forbidden; other points on the unit circle have little effect other than to prevent the base iteration from converging.) Figure 4 shows the results of cycling MPE and VEA with $k = 2$, 4, 6, and 8 (bottom to top, respectively), to illustrate that one need not correctly detect k in order to get a highly accurate answer. For $k < 8$, the two methods are very similar (with a slight advantage to VEA); both converge linearly, and at a rate dependent on k. For $k = 8$, MPE needs the second cycle to achieve machine accuracy, which offsets its efficiency in use of base iterates.

Example 5. *Large negative eigenvalues.* Here we use the 10 by 10 problem generator again, with $c_1 = 0.001$, $c_2 = 0.4$, $c_3 = 0.2$, $c_4 = 0.1$, $c_5 = 0.9$. Thus the dominant eigenvalues are a pair at -0.9. The results for all four methods ($k = 8$) are shown in Fig. 5. Note that (a) MPE and RRE are virtually identical and little different from VEA; (b) MPE and RRE need *three* cycles to reach their maximum accuracy; (c) SEA is slightly better than VEA on the first cycle, but is not improved by cycling. The deterioration of SEA on a second cycle has been observed with some other examples as well, but with

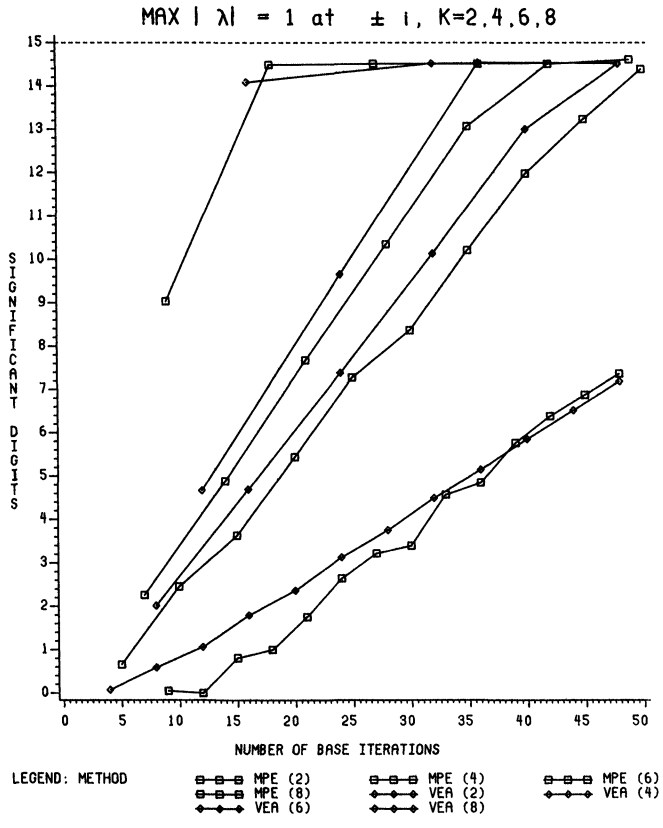

FIG. 4. *Divergent* 10 *by* 10.

no obvious pattern. For example, when SEA is applied to the five cases in Example 3, the second cycle shows no change for $c_1 = 0.9$, marked deterioration for $c_1 = 0.99$, slight improvement for $c_1 = 0.999$ and 0.9999, and marked improvement for $c_1 = 0.99999$.

Example 6. *Nonlinear iteration.* Our first nonlinear example is the determination of a dominant eigenvector by a variant of the power method. Specifically, each iterate is computed from the previous one by multiplication by the given matrix B and scalar division by the first component

$$F(\mathbf{x}) = B\mathbf{x}/(B\mathbf{x})_1.$$

We use

$$B = \begin{bmatrix} 3.4 & -3.7 & 2.4 & -0.6 \\ 2.4 & -2.5 & 2.2 & -0.6 \\ 2.4 & -3.6 & 3.6 & -0.9 \\ 2.8 & -5.2 & 4.8 & -0.9 \end{bmatrix},$$

$$\mathbf{x}_0 = (2, 1, 0.5, 2)^t.$$

The dominant eigenvalue is 1.5, and the corresponding eigenvector is $\mathbf{s} = (1, 1, 1, 1)^t$. The power sequence diverges, but the base iteration converges at a moderate rate: The eigenvalues of $F'(\mathbf{s})$ are: 0.533, 0.467, 0.4, and 0.

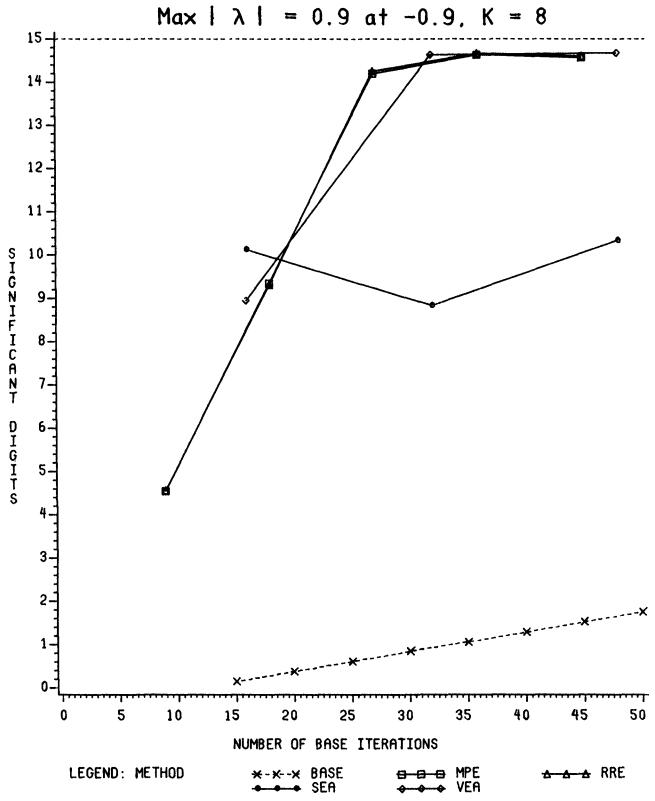

FIG. 5. *Convergent* 10 *by* 10.

The results for this problem are shown in Fig. 6, which is typical of slowly convergent, nonlinear iterations with "dominant" linear part (except for the apparent failure of SEA after the first cycle): The quadratic convergence (final term of each cycle) is evident, up to machine accuracy. MPE and RRE are shown with $k=3$, the proper degree because the first component of each vector was required to be unity. The epsilon methods were allowed to run to $k=4$ because the vectors had length four.

We digress here to comment on the application of these method to the infamous Bodewig matrix [3], a 4 by 4 integer matrix for which the power method requires some 800 iterations to get anywhere near an eigenvector for the dominant eigenvalue. The problem arises from roots that are close in magnitude and opposite in sign. (-8.03 and 7.93), and the power sequence is initially "close" to an eigenvector for the smaller root, 7.93 (1 to 2 SD accuracy in each component at 20 terms). All four methods find this latter eigenvector to machine accuracy, with numbers of base iterations and significance levels very similar to those shown in Fig. 6. It is of interest to note that such a sequence that eventually converges to a very different vector actually "contains" this information.

Example 7. *Quadratic iteration.* Gekeler [17] illustrates the vector epsilon algorithm with a number of quadratic iterations that can be written in the form

$$F(\mathbf{x}) = \mathbf{b} + A\mathbf{x} + Q(\mathbf{x}), \tag{11.2}$$

where $Q(\mathbf{x})$ is the quadratic part. All of these are 4-dimensional and all have been constructed to have solution $(1, 1, 1, 1)^t$. However, they have other solutions as well, and depending on the starting point, some of the base iterations and/or epsilon

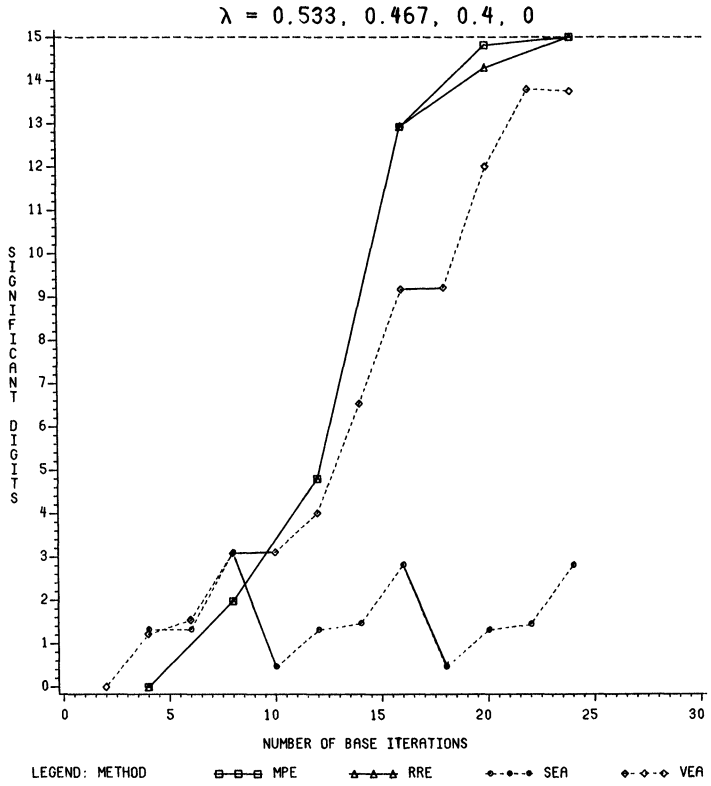

FIG. 6. *Normalized power method (nonlinear).*

accelerations converge to other solutions. Unfortunately, Gekeler does not indicate in each case what solution was actually found.

We will discuss two of these problems [17, Examples I, V], the first of which has

$$A=\begin{bmatrix} 2.25 & 0.01 & 0.05 & 0.5 \\ 0.01 & 1.75 & 0. & 0.05 \\ 0.05 & 0. & 1.75 & 0.01 \\ 0.5 & 0.05 & 0.01 & 2.25 \end{bmatrix},$$

$$\mathbf{b}=(-0.81,\ -0.31,\ -0.31,\ -0.81)^t,$$

$$Q(\mathbf{x})=-0.5\left(x_1^2+x_1x_4,\ x_2^2,\ x_3^2,\ x_1x_4+x_4^2\right)^t,$$

$$\mathbf{x}_0=(2,2,2,2)^t.$$

The eigenvalues of $F'(\mathbf{s})$ are 0.9, 0.8, 0.7, and 0.6. The results for this problem (Fig. 7) show again that MPE and RRE do not always retain their 2-to-1 efficiency in use of base iterates. (RRE and SEA are not shown in Fig. 7, but their results were similar, respectively, to MPE and VEA.) The figure shows VEA both for $k=4$ and $k=3$ to illustrate the point that often it makes little difference whether the correct k is detected exactly.

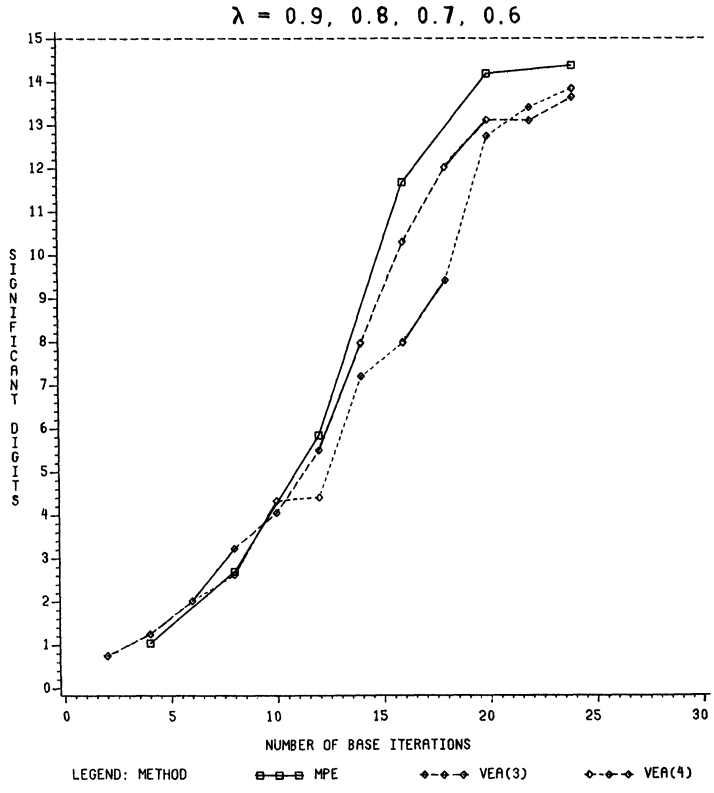

FIG. 7. *Quadratic sequence generator.*

Example 8. *Quadratic iteration with two solutions.* Gekeler's Example V in [17] has the form (11.2) with

$$A = \begin{bmatrix} 3.9 & -3.7 & 2.4 & -0.6 \\ 2.4 & -2.0 & 2.2 & -0.6 \\ 2.4 & -3.6 & 4.1 & -0.9 \\ 2.8 & -5.2 & 4.8 & -0.4 \end{bmatrix},$$

$$\mathbf{b} = -0.75(1, 1, 1, 1)^t,$$

$$Q(\mathbf{x}) = -0.25(x_1^2, x_2^2, x_3^2, x_4^2)^t,$$

$$\mathbf{x}_0 = 1.5(1, 1, 1, 1)^t.$$

The iteration has both $(1, 1, 1, 1)^t$ and $(3, 3, 3, 3)^t$ as fixed points. The base iteration, VEA, and SEA all converge to the latter from the given starting point, but MPE initially jumps to components less than unity and then approaches $(1, 1, 1, 1)^t$ from below. Figure 8 shows roughly quadratic convergence of MPE with $k=2$ and VEA with $k=4$. But this convergence is to *different* answers, and thus is governed by different sets of eigenvalues. In fact, $F'(1, 1, 1, 1)$ is the matrix B in our Example 6, which has eigenvalues 1.5, 0.8, 0.7, 0.6, whereas $F'(3, 3, 3, 3) = B - I$, which has eigenvalues 0.5, -0.4, -0.3, -0.2. Thus MPE is solving a divergent problem with all the eigenvalues relatively large, while VEA is solving an "easier" problem with relatively small eigenvalues. In particular, the relevant set of eigenvalues is misstated in [17].

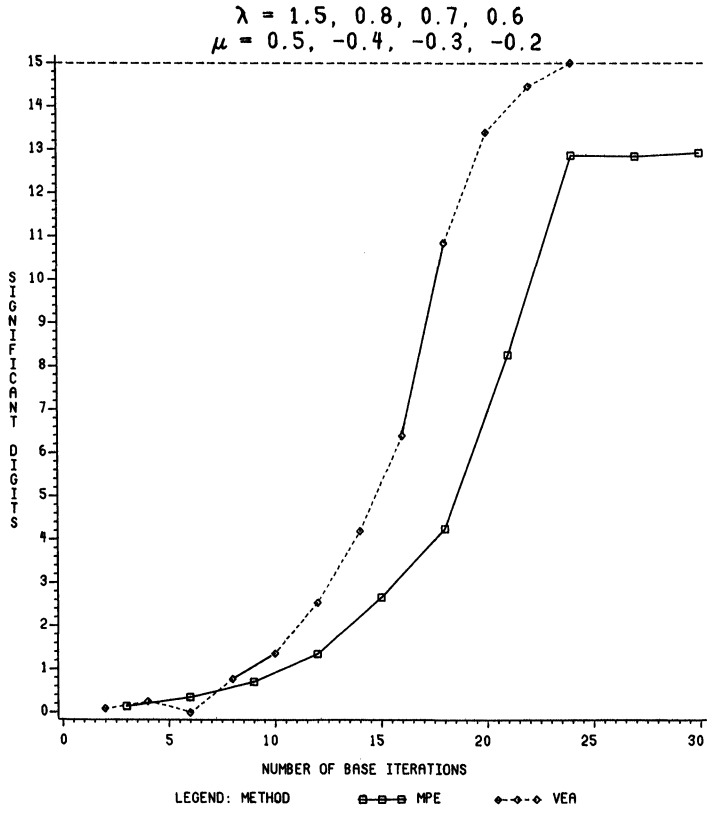

FIG. 8. *Quadratic generator, 2 solutions.*

This example is another for which RRE gives results very similar to MPE when coded in APL, but fails to converge when coded in FORTRAN. The SEA results were nearly identical to those for VEA with both programs.

12. Summary. We have motivated and derived five nonlinear extrapolation methods for vector sequences with a limit or an anti-limit and for which the sequence generator may not be known explicitly: the minimal polynomial extrapolation (MPE), the reduced rank extrapolation (RRE), and the topological, scalar, and vector epsilon algorithms (TEA, SEA, and VEA). All five give an exact extrapolation if the sequence is generated linearly and if they are applied to a system of equations of the right rank k (the degree of a minimal polynomial of the sequence generator), which is often smaller than the dimension of the space in which the vectors are generated.

All but TEA can also be implemented without knowing the correct dimension k, achieving approximate extrapolation if k is the number of dominant eigenvalues of the sequence generator (or of its linear part if it is not linear). This value of k often can be determined empirically with a reasonable amount of computation, and the extrapolations can be iterated ("cycled") in the same dimension to obtain a sequence of approximations that often converges quadratically. Since attempts to implement TEA this way do not appear to work, its significance remains as a theoretical bridge between the minimal polynomial methods (MPE and RRE) and the epsilon methods (SEA and VEA).

Our numerical tests suggest some tentative guidelines for practical use of these methods. Numerical difficulties anticipated on theoretical grounds for RRE and SEA do occur, and we see little or no reason ever to prefer RRE to MPE or SEA to VEA. MPE has a theoretical advantage over VEA in that, under conditions in which the quadratic convergence "theorem" holds, it should achieve the same accuracy from roughly half as many base iterates (steps of the sequence generator) and with less storage. However, quadratic convergence of MPE is more likely to fail, and the algorithm itself is more sensitive to certain numerical effects, such as rapid divergence of the base sequence. Thus there are circumstances in which either MPE or VEA might give the better result for a given problem of the type considered, and those circumstances are difficult to determine in advance. Both methods can sometimes be enhanced by intercycle runs of the sequence generator and/or sampling the base sequence at regular intervals instead of using consecutive terms.

Acknowledgments. We are grateful to Professor Claude Brezinski for encouragement of this work and many helpful conversations, and to Dr. R. P. Eddy for his helpful communications by mail. We are further indebted to two anonymous referees of an earlier draft by the first two authors; their comments led to a much more substantial paper. All computations reported here were done in FORTRAN double precision on an IBM 3081 at Triangle Universities Computation Center with programs written by Amy Dunham (adapted in part from PL/I programs by David Smith II). Least squares problems were solved with the IMSL routine LLSQF.

REFERENCES

[1] A. C. AITKEN, *The evaluation of the latent roots and latent vectors of a matrix*, Proc. Roy. Soc. Edinburgh, 57 (1936–37), pp. 269–304.

[2] J. BEUNEU, *Minimal polynomial extrapolation methods*, preprint, 1985.

[3] E. BODEWIG, *A practical refutation of the iteration method for the algebraic eigenvalue problem*, MTAC, 8 (1954), pp. 237–239.

[4] C. BREZINSKI, *Application de l'ε-algorithme à la résolution des systèmes non linéaires*, C. R. Acad. Sci. Paris, Sér. A, 271 (1970), pp. 1174–1177.

[5] _____, *Sur un algorithme de résolution des systèmes non linéaires*, C. R. Acad. Sci. Paris, Sér. A, 272 (1971), pp. 145–148.
[6] _____, *Some results in the theory of the vector ε-algorithm*, Linear Alg. Appl., 8 (1974), pp. 77–86.
[7] _____, *Computation of the eigenelements of a matrix by the ε-algorithm*, Linear Alg. Appl., 11 (1975), pp. 7–20.
[8] _____, *Généralizations de la transformation de Shanks, de la table de Padé, et de l'ε-algorithme*, Calcolo, 12 (1975), pp. 317–360.
[9] _____, *Accélération de la convergence en analyse numérique*, Lecture Notes in Mathematics 584, Springer-Verlag, Berlin, 1977.
[10] _____, *Algorithmes d'accélération de la convergence: étude numérique*, Editions Technip, Paris 1978.
[11] C. BREZINSKI AND A. C. RIEU, *The solution of systems of equations using the ε-algorithm, and an application to boundary-value problems*, Math. Comp., 28 (1974), pp. 731–741.
[12] S. CABAY AND L. W. JACKSON, *A polynomial extrapolation method for finding limits and antilimits of vector sequences*, SIAM J. Numer. Anal., 13 (1976), pp. 734–752.
[13] R. P. EDDY, *Causes and countermeasures for ping-pong oscillations in the output of the* DX-DXG FFLS DD07 *computer program* (Appendix A. Mathematical Theory of Iteration), Applied Mathematics Laboratory Technical Note AML-35-68, Naval Ship Research and Development Center, Washington, DC, 1968.
[14] _____, *Extrapolating to the limit of a vector sequence*, in Information Linkage Between Applied Mathematics and Industry, P. C. C. Wang, ed., Academic Press, New York, 1979, pp. 387–396.
[15] D. K. FADDEEV AND V. N. FADDEEVA, *Computational Methods of Linear Algebra*, W. H. Freeman, San Francisco, 1963.
[16] L. FOX AND E. T. GOODWIN, *The numerical solution of nonsingular integral equations*, Philos. Trans. Roy. Soc. London, Ser. A 245 (1953), pp. 501–534.
[17] E. GEKELER, *On the solution of systems of equations by the epsilon algorithm of Wynn*, Math. Comp. 26 (1972), pp. 427–436.
[18] B. GERMAIN-BONNE, *Estimation de la limite de suites et formalization de procédés d'accélération de convergence*, Thèse, Université des Sciences et Techniques de Lille, 1978.
[19] P. R. GRAVES-MORRIS, *Vector valued rational interpolants* I, Numerische Mathematik, 42 (1983), pp. 331–348.
[20] R. T. GREGORY AND D. L. KARNEY, *A Collection of Matrices for Testing Computational Algorithms*, Wiley-Interscience, New York, 1969.
[21] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.
[22] P. HENRICI, *Elements of Numerical Analysis*, John Wiley, New York, 1964.
[23] C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
[24] R. LUDWIG, *Verbesserung einer Iterationsfolge bei gleichungssystemen*, Z. Angew. Math. Mech., 32 (1952), pp. 232–234.
[25] J. B. MCLEOD, *A note on the ε-algorithm*, Computing, 7 (1971), pp. 17–24.
[26] M. MEŠINA, *Convergence acceleration for the iterative solution of the equations* $X=AX+f$, Comput. Methods Appl. Mech. Engrg., 10 (1977), pp. 165–173.
[27] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.
[28] R. J. SCHMIDT, *On the numerical solution of linear simultaneous equations by an iterative method*, Phil. Mag. 7, 32 (1941), pp. 369–383.
[29] D. SHANKS, *Non-linear transformations of divergent and slowly convergent sequences*, J. Math. Phys., 34 (1955), pp. 1–42.
[30] A. SIDI, W. F. FORD AND D. A. SMITH, *Acceleration of convergence of vector sequences*, SIAM J. Numer. Anal., 23 (1986), pp. 178–196.
[31] S. SKELBOE, *Computation of the periodic steady-state response of nonlinear networks by extrapolation methods*, IEEE Trans. Circuits and Systems, 27 (1980), pp. 161–175.
[32] G. STRANG, *Linear Algebra and its Applications*, Academic Press, New York, 1976.
[33] Y. V. VOROBYEV, *Method of Moments in Applied Mathematics*, Gordon and Breach, New York, 1965.
[34] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1969.
[35] P. WYNN, *On a device for computing the* $e_m(S_n)$ *transformation*, MTAC, 10 (1956), pp. 91–96.
[36] _____, *Acceleration techniques for iterated vector and matrix problems*, Math. Comp., 16 (1962), pp. 301–322.
[37] _____, *Continued fractions whose coefficients obey a noncommutative law of multiplication*, Arch. Rat. Mech. Anal., 12 (1963), pp. 273–312.