Solving Linear Equations by Extrapolation

Walter Gander[†] , Gene H. Golub[‡] and Dominik Gruntz[†]

Abstract

This is a survey paper on extrapolation methods for vector sequences. We have simplified some derivations and we give some numerical results which illustrate the theory.

1 Introduction

Many iterative methods have been proposed for solving large systems of equations which are either sparse or structured or both. Unfortunately, the iteration procedures converge rather slowly and consequently acceleration methods are required. If the original matrix is symmetric and positive definite, then the conjugate gradient method can be used in combination with the Jacobi or SSOR iteration method. Such procedures often have excellent convergence properties.

For non-symmetric problems there is a great need to develop, analyze and experiment with a variety of methods. The solution time is not only a function of the convergence properties but of the architectural structure of the computer being used. To the best of our knowledge, there is no acceleration procedure which is uniformly best for all architectures.

In [2], a system of equations was given; the efficiency of the ε-algorithm was demonstrated in [4] in extrapolating the solution from approximations obtained from Block Gauss-Seidel iterations. Though the Gauss-Seidel method is convergent for diagonally dominant matrices cf. [18], convergence may be very slow.

In this paper, we continue our study and show to the relevance of this and other methods. All methods make use of *the minimal polynomial of a matrix with respect to a vector*. Since this polynomial is not known in practice, there are several ways to approximate it. This leads to the algorithms *minimal polynomial extrapolation* related to the *method of Arnoldi*, *reduced rank extrapolation* related to *GCR* or *orthomin* and *topological ε-algorithm* related to *Lanczos*. We also mention the algorithms *modified minimal polynomial extrapolation*, *scalar-* and *vector ε-algorithm* which are not related to known Krylov subspace methods. Finally we report some numerical results from experiments with large linear systems obtained from the discretisation of an elliptic differential equation.

Much of the theoretical work on extrapolation methods has been done by Brezinski [1] and his students [3] and by Avram Sidi et al. An extensive list of publications are given in the **References**.

2 Notation and general comments to the methods

Given a matrix A and a vector b, we consider the vector sequence generated in real or complex n-space from a starting vector x_0 by

$$x_{j+1} = Ax_j + b, \qquad j = 0, 1, \ldots \tag{1}$$

It is assumed that 1 is not an eigenvalue of A and therefore that the iteration has a unique fixed point

$$s = (I - A)^{-1}b, \tag{2}$$

which is the solution of the linear system $Cx = b$ with $C = I - A$.

If all the eigenvalues of A are less than one in magnitude, then $s = \lim_{j\to\infty} x_j$ for all x_0.

If the sequence diverges, s is called the *anti-limit* and one may still try to determine s from a finite number of terms of the sequence.

Let $e_j := x_j - s$ denote the error and assume $A = QDQ^{-1}$ where Q is the matrix of the eigenvectors q_j and $D = diag\{\lambda_1, \ldots, \lambda_n\}$ the eigenvalues of A. Then from iteration (1), we have

$$e_j = A^j e_0 = QD^j \underbrace{Q^{-1}e_0}_{z} = \sum_{l=1}^{n} q_l \lambda_l^j z_l \tag{3}$$

[†]Institut für wissenschaftliches Rechnen, Eidgenössische Technische Hochschule, CH-8092 Zürich, Switzerland, e-mail : gander@inf.ethz.ch respectively gruntz@inf.ethz.ch.

[‡]Department of Computer Science, Stanford University, Stanford, CA 94305, e-mail : na.golub@na-net.stanford.edu. The work of this author was supported in part by the National Science Foundation under grand DCR-8412314.

NATO ASI Series, Vol. F 62
Supercomputing
Edited by J. S. Kowalik
© Springer-Verlag Berlin Heidelberg 1990

and therefore

$$x_j = s + \sum_{l=1}^{n} q_l \lambda_l^j z_l. \tag{4}$$

The idea of extrapolation is to form a *linear combination of the x_j, so that error terms in (4) are cancelled.* We wish to find $\gamma_0, \ldots, \gamma_k$ such that

$$\begin{aligned} \textstyle\sum_{j=0}^{k} \gamma_j &= 1 \\ \textstyle\sum_{j=0}^{k} \gamma_j x_j &= \textstyle\sum_{j=0}^{k} \gamma_j s = s. \end{aligned} \tag{5}$$

Now taking the same linear combination with the e_j in (3) gives

$$0 = \sum_{j=0}^{k} \gamma_j e_j = \sum_{j=0}^{k} \gamma_j A^j e_0 = P_k(A) e_0.$$

P_k is some polynomial with coefficients γ_j such that $P_k(A)e_0 = 0$. Note that (5) implies $P_k(I) = I$. Of course, one wishes to have k as small as possible. Therefore we *choose as γ_j the coefficients of the minimal polynomial of A with respect to e_0 with the normalization $\sum_{j=0}^{k} \gamma_j = 1$, $\gamma_k \neq 0$.*

If $P_k(A)e_0 = 0$ then by multiplying with A^m one obtains

$$A^m P_k(A) e_0 = P_k(A) A^m e_0 = P_k(A) e_m = 0.$$

Therefore if A is non singular then P_k is also minimal polynomial of A with respect to e_m for all m (cf. lemma 1 in [15]).

It is not possible to compute P_k since the error e_m is unknown. However let

$$u_j = \Delta x_j = x_{j+1} - x_j \tag{6}$$

denote the difference of two successive iterates. Then we have the important relation

$$u_j = (A - I) e_j = -C e_j. \tag{7}$$

Note that for the *residual vector,*

$$r_j = b - C x_j = A x_j + b - x_j = x_{j+1} - x_j = u_j \tag{8}$$

therefore the differences of successive iterates are also the residuals.

Consider now the minimal polynomial of A with respect to u_m:

$$P_k(A) u_m = 0. \tag{9}$$

Because $u_m = (A - I)e_m$ and $(A - I)$ is nonsingular and commutes with $P_k(A)$, the minimal polynomial with respect to e_m is also P_k, since

$$P_k(A)(A - I)e_m = (A - I)P_k(A)e_m = 0 \iff P_k(A)e_m = 0.$$

Let $P_k(\lambda) = c_0 + c_1\lambda + \cdots + c_k\lambda^k$ with $c_k \neq 0$. Then

$$\sum_{j=0}^{k} c_j (x_{m+j} - s) = \sum_{j=0}^{k} c_j x_{m+j} - \sum_{j=0}^{k} c_j s = 0,$$

and we get the barycentric representation

$$s = \frac{\sum_{j=0}^{k} c_j x_j}{\sum_{j=0}^{k} c_j}. \tag{10}$$

Since we assumed that $A - I$ is non singular, $\lambda = 1$ is not an eigenvalue of A and therefore $P_k(1) = \sum_{j=0}^{k} c_j \neq 0$. *We can therefore extrapolate s from $x_m, \ldots, x_{m+k}$ by using the coefficients of the minimal polynomial of A with respect to u_m.*

If we define the coefficients

$$\gamma_j := c_j / \sum_{i=0}^{k} c_i, \tag{11}$$

we can also write the barycentric representation as given in equation (5).

We conclude this section with some definitions which will be used in the following.

$$w_j = \Delta^2 x_j = \Delta u_j = u_{j+1} - u_j. \tag{12}$$

For some fixed k, we introduce the matrices

$$U = [u_m, u_{m+1}, \dots, u_{m+k-1}] \tag{13}$$

$$W = [w_m, w_{m+1}, \dots, w_{m+k-1}]. \tag{14}$$

We note that for each j,

$$u_j = Au_{j-1} = A^j u_0. \tag{15}$$

The same relation holds between w_j and u_j as between u_j and e_j:

$$w_j = (A - I)u_j. \tag{16}$$

Writing this relation, using the matrices U and W, we get

$$W = (A - I)U. \tag{17}$$

3 Minimal Polynomial Extrapolation (MPE)

Let P_k be the (unique) minimal polynomial of A of degree k with respect to u_m, i.e.

$$P_k(A)u_m = 0. \tag{18}$$

Let $P_k(\lambda) = c_0 + c_1\lambda + \cdots + c_k\lambda^k$ with $c_k = 1$. Using (15), equation (18) becomes

$$\sum_{j=0}^{k-1} c_j u_{m+j} = -u_{m+k}. \tag{19}$$

In [15] it is proposed to solve (19) as linear least squares problem for c_j (numerically this may not be a good way to go, see p. 371 in [19]). The normal equations are

$$U^T U c = -U^T u_{m+k}$$

$$\iff \sum_{j=0}^{k-1} (u_{m+i}, u_{m+j})c_j = -(u_{m+i}, u_{m+k}) \quad i = 0, \dots, k-1.$$

Using the fact that $c_k = 1$, we divide the equations by $\sum_{j=0}^{k} c_j$ and obtain with γ_j from (11)

$$\begin{array}{rcl} \sum_{j=0}^{k}(u_{m+i}, u_{m+j})\gamma_j & = & 0 \quad i = 0, \dots, k-1 \\ \sum_{j=0}^{k} \gamma_j & = & 1. \end{array} \tag{20}$$

Since k, the degree of the minimal polynomial, is in general unknown using the coefficients computed by (20) will give only an approximation to s:

$$s_{mk} = \sum_{j=0}^{k} \gamma_j x_{m+j}. \tag{21}$$

One can show (cf. [11]) that for k values smaller than the degree of the minimal polynomial, solutions of (20) exist if the matrix $U^T C U$ is non singular, that is, if the symmetric part of C is positive definite. In the same paper a connection to the method of Arnoldi is established.

In the method of Arnoldi an orthogonal basis is constructed from the Krylov subspace

$$\mathcal{K}_k(C, r_0) := \text{span}\{r_0, Cr_0, \dots, C^{k-1}r_0\}.$$

Using Gram-Schmidt, one computes the QR-decomposition of the matrix

$$[r_0, Cr_0, \dots, C^{k-1}r_0] = P_k R.$$

Let p_j denote the j-th column of the orthogonal matrix P_k. Then $p_0 = r_0/||r_0||$ and

$$h_{j+1,j}p_{j+1} = Cp_j - \sum_{i=0}^{j} h_{ij}p_i \tag{22}$$

with $h_{ij} = p_i^T Cp_j$ and $h_{j+1,j}$ such that $||p_{j+1}|| = 1$. From equation (22) we conclude that

$$CP_k = P_k H_k + p_{k+1}e_k^T h_{k+1,k} \tag{23}$$

where H_k is a Hessenberg matrix. To compute an approximate solution to the linear system $Cx = b$, one makes the *ansatz*

$$x_k = x_0 + P_k y \quad \Rightarrow r_k = r_0 - CP_k y \in \mathcal{K}_{k+1}(C, r_0)$$

and determines y such that

$$r_k \perp \mathcal{K}_k(C, r_0) \iff H_k y = ||r_0|| e_1.$$

On the other hand, if we compute the residual of (21) with $m = 0$ then

$$r(s_{0k}) = \sum_{j=0}^{k} \gamma_j r(x_j) = \sum_{j=0}^{k} \gamma_j A^j u_0 \in \mathcal{K}_{k+1}(A, r_0).$$

Now $\mathcal{K}_{k+1}(A, r_0) = \mathcal{K}_{k+1}(C, r_0)$ since $I - A = C$. Furthermore from the normal equation (20) we see that

$$r(s_{mk}) \perp \mathcal{K}_k(A, r_0) = \mathcal{K}_k(C, r_0).$$

Therefore,

Theorem 1 *The residuals of the approximations of the method of Arnoldi and of MPE are in the same subspaces and thus the two methods are equivalent.*

4 Reduced rank extrapolation (RRE)

In the method of Arnoldi, the approximate solution x_k is formed as a linear combination of columns the orthogonal matrix P_k, which form an orthogonal basis of the Krylov-subspace $\mathcal{K}_k(C, r_m)$

$$s_{mk} = x_m + P_k y \in \mathcal{K}_k(C, r_m) = \mathcal{K}_k(A, u_m).$$

Also the columns of $U = [u_m, \dots, u_{m+k-1}]$ are a basis of $\mathcal{K}_k(A, u_m)$ and therefore one can also express the approximate solution as

$$s_{mk} = x_m + U\xi. \tag{24}$$

If k is the degree of the minimal polynomial of A with respect to u_m then $s_{mk} = s$. Therefore we have to find ξ such that

$$U\xi = -e_m = s - x_m. \tag{25}$$

However since e_m is not known, this equation is not practical. By multiplying (25) from the left by $A - I$, we get a computable right hand side:

$$W\xi = (A - I)U\xi = -(A - I)e_m = -u_m. \tag{26}$$

This equation defines the

Reduced rank extrapolation (RRE) method

- generate $x_m, x_{m+1}, \ldots, x_{m+k+1}$
- compute $U = [u_m, u_{m+1}, \ldots, u_{m+k-1}]$ and $u_{m+k} = x_{m+k+1} - x_{m+k}$
- compute $W = [w_m, w_{m+1}, \ldots, w_{m+k-1}]$ where $w_j = u_{j+1} - u_j$
- Solve the least squares problem $W\xi = -u_m$ so that $\xi = -W^+ u_m$
- compute $s_{mk} = x_m + U\xi$

Writing ΔX for U and $\Delta^2 X$ for W, equation (24) becomes

$$s_{mk} = x_m - \Delta X(\Delta^2 X)^+ \Delta x_m$$

and the analogy of reduced rank extrapolation and Aitken's Δ^2 method becomes visible.

The normal equations of (26) are

$$W^T W\xi = -W^T u_m$$

$$\iff \sum_{j=0}^{k-1} (w_{m+i}, w_{m+j})\xi_j = -(w_{m+i}, u_m) \quad i = 0, \ldots, k-1.$$

If we replace $w_{m+j} = u_{m+j+1} - u_{m+j}$, we obtain

$$(w_{m+i}, u_m)(1 - \xi_0) + \sum_{j=1}^{k-1} (w_{m+i}, u_{m+j})(\xi_{j-1} - \xi_j) + (w_{m+i}, u_{m+k})\xi_{k-1} = 0.$$

We now introduce the new variables

$$\begin{aligned} \gamma_0 &= 1 - \xi_0 \\ \gamma_j &= \xi_{j-1} - \xi_j \quad j = 1, \ldots, k-1 \\ \gamma_k &= \xi_{k-1}. \end{aligned} \tag{27}$$

Notice there are $k+1$ variables γ_j and only k variables ξ_j, however the γ_j are related by the relationship $\sum_{j=0}^{k} \gamma_j = 1$. Expressed in this new variables the normal equations become

$$\begin{aligned} \textstyle\sum_{j=0}^{k} (w_{m+i}, u_{m+j})\gamma_j &= 0 \\ \textstyle\sum_{j=0}^{k} \gamma_j &= 1. \end{aligned} \tag{28}$$

Furthermore

$$\begin{aligned} s_{mk} &= x_m + \textstyle\sum_{j=0}^{k-1} u_{m+j}\xi_j \\ &= x_m + \textstyle\sum_{j=0}^{k-1} (x_{m+j+1} - x_{m+j})\xi_j \\ &= x_m(1 - \xi_0) + \textstyle\sum_{j=1}^{k-1} x_{m+j}(\xi_{j-1} - \xi_j) + x_{m+k}\xi_{k-1} \end{aligned}$$

and therefore again,

$$s_{mk} = \sum_{j=0}^{k} x_{m+j}\gamma_j \quad \text{with} \quad \sum_{j=0}^{k} \gamma_j = 1.$$

If k is smaller than the degree of the minimal polynomial, then in contrast to MPE the intermediate solutions always exists for RRE [11]. RRE is connected to GCR or ORTHOMIN.

ORTHOMIN or GCR algorithm:

$$\begin{aligned} r_0 &= b - Cx_0; \quad p_0 = r_0 \\ \alpha_j &= \frac{(r_j, Cp_j)}{(Cp_j, Cp_j)} & \Rightarrow r_{j+1} \perp Cr_j \\ x_{j+1} &= x_j + \alpha_j p_j \\ r_{j+1} &= r_j - \alpha_j Cp_j \\ \beta_{ij} &= \frac{(Cr_{j+1}, Cp_i)}{(Cp_i, Cp_i)}, \quad i = 0, \ldots, j & \Rightarrow Cp_{j+1} \perp Cp_i \\ p_{j+1} &= r_{j+1} - \textstyle\sum_{i=0}^{j} \beta_{ij} p_i. \end{aligned}$$

This algorithm has the following properties:

1. $r_k \in \mathcal{K}_{k+1}(C, r_0)$
2. $r_k \perp Cr_j, \quad j < k$ ("conjugate residuals")
3. $r_k \perp \mathcal{K}_k(C, Cr_0)$.

For the residuals in RRE we have

$$r(s_{mk}) = \sum_{j=0}^{k} \underbrace{r(x_{m+j})}_{u_{m+j}} \gamma_j \in \mathcal{K}_{k+1}(A, u_m) = \mathcal{K}_{k+1}(C, r_m).$$

It follows from the normal equations (28) that

$$r(s_{mk}) \perp w_{m+j}, \quad \text{for} \quad i = 1, \ldots, k-1.$$

Since $w_{m+j} = A^j w_m$ this is equivalent to $r(s_{mk}) \perp \mathcal{K}_k(A, w_m)$. Finally since $w_m = (A - I)u_m = -Cu_m$, we have

$$\mathcal{K}_k(A, w_m) = \mathcal{K}_k(A, Cu_m) = \mathcal{K}_k(C, Cu_m) = \mathcal{K}_k(C, Cr_m).$$

Now for $m = 0$, $r(s_{0k}) \in \mathcal{K}_{k+1}(C, r_0)$ and $r(s_{0k}) \perp \mathcal{K}_k(C, Cr_0)$. Hence

Theorem 2 *The residuals of RRE are in the same subspaces as those of GCR. The methods are therefore equivalent.*

5 Modified minimal polynomial extrapolation (MMPE)

Instead of solving the system (26) $W\xi = -u_m$ in the least squares sense, an arbitrary matrix $Q \in \mathrm{R}^{n \times k}$ with full rank k is chosen and the solution ξ is computed from the $k \times k$ system

$$Q^T W \xi = -Q^T u_m.$$

Writing $Q = [q_0, \ldots, q_{k-1}]$, these equations are

$$\sum_{j=0}^{k-1} (q_i, w_{m+j}) \xi_j = -(q_i, w_m), \quad i = 0, \ldots, k-1.$$

Introducing the new variables γ_j we obtain by the same manipulation as in the previous section the equations

$$\begin{array}{rcl} \sum_{j=0}^{k} (q_i, u_{m+j}) \gamma_j & = & 0 \\ \sum_{j=0}^{k} \gamma_j & = & 1. \end{array} \qquad (29)$$

Note that the subspace spanned by the columns of Q can be interpreted as a Krylov subspace (with some matrix G). We construct G as follows: Augment linear independent column vectors $\tilde{q}_j$ to form a $n \times n$ non singular matrix

$$\tilde{Q} = [q_0, \ldots, q_{k-1}, \tilde{q}_k, \ldots, \tilde{q}_n].$$

Let us denote $\tilde{q}_j = q_j, \quad j = 0, \ldots, k-1$. We wish to find G such that $\tilde{q}_{j+1} = G\tilde{q}_j$ i.e.

$$[\tilde{q}_1, \ldots, \tilde{q}_n, x] = G\tilde{Q}$$

where x is an arbitrary vector. From this equation we get

$$G = [\tilde{q}_1, \ldots, \tilde{q}_n, x]\tilde{Q}^{-1}.$$

Therefore we have an analogous result as with RRE:

$$r(s_{mk}) \in \mathcal{K}_{k+1}(A, u_m) \quad \text{and} \quad r(s_{mk}) \perp span\{q_0, \ldots, q_{k-1}\} = \mathcal{K}_k(G, q_0).$$

Taking $q_j = e_i$ with randomly chosen unit vectors such that rank $Q = k$ (i.e. picking simply k equations out of $W\xi = -u_m$) minimizes the computational work. If k is smaller than the degree of the minimal polynomial then it is shown in [11], that the intermediate values s_{mk} exists if $Q^T CU$ is non singular.

6 The topological ε-algorithm (TEA)

Here again one replaces the linear system (26) $W\xi = -u_m$ by $Q^T W\xi = -Q^T u_m$. However now

$$Q = [q, A^T q, \dots, (A^T)^{k-1} q] \tag{30}$$

and q is some fix chosen vector. By the same calculation as before we obtain the equations for γ_j, namely,

$$0 = \sum_{j=0}^{k}((A^T)^i q, u_{m+j})\gamma_j = \sum_{j=0}^{k}(q, A^i u_{m+j})\gamma_j = \sum_{j=0}^{k}(q, u_{m+j+i})\gamma_j, \quad (i = 0, \dots, k-1).$$

We note that for this extrapolation, we need the iterates $x_m, \dots, x_{m+2k}$ whereas for MPE,RRE and MMPE we only need $x_m, \dots, x_{m+k+1}$. Originally TEA was developed by Brezinski in analogy to Wynn's ε-algorithm for extrapolation of sequences in a topological vector space. The approximations can be computed recursively without using the normal equations cf. section 9. It is interesting that TEA is connected [11] to the non-symmetric Lanczos algorithm [7].

Non-symmetric Lanczos

```
Choose $y_0$ and $z_0$ such that $y_0^T z_0 = 1$
for $j = 0, 1, \dots, k$ do
    $\alpha_j = (Cy_j, z_j)$
    $\hat{y}_{j+1} = Cy_j - \alpha_j y_j - \beta_j y_{j-1}$   $(\beta_0 y_{-1} = 0)$
    $\hat{z}_{j+1} = C^T z_j - \alpha_j z_j - \delta_j z_{j-1}$   $(\delta_0 z_{-1} = 0)$
    Choose $\delta_{j+1}$ and $\beta_{j+1}$ such that $\delta_{j+1}\beta_{j+1} = \hat{y}_{j+1}^T \hat{z}_{j+1}$
      e.g. $\delta_{j+1} = \sqrt{|\hat{y}_{j+1}^T \hat{z}_{j+1}|}$ and $\beta_{j+1} = sign(\hat{y}_{j+1}^T \hat{z}_{j+1})\delta_{j+1}$
    $y_{j+1} = \hat{y}_j / \delta_{j+1}$
    $z_{j+1} = \hat{z}_j / \beta_{j+1}$.
```

This algorithm has the following properties:

1. $Y_k^T Z_k = I$ (the vector y_j and z_i are *biorthogonal*)
2. $\hat{y}_{j+1} = P_j(C) y_0$ and $\hat{z}_{j+1} = P_j(C^T) z_0$, i.e. $\hat{y}_j \in \mathcal{K}_j(C, y_0)$ and $\hat{z}_j \in \mathcal{K}_j(C^T, z_0)$
3. $$Z_k^T C Y_k = T_k := \begin{pmatrix} \alpha_0 & \beta_1 & & \\ \delta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \delta_k & \alpha_k \end{pmatrix}$$
4. $CY_k = Y_k T_k + \underbrace{\delta_{k+1} y_{k+1}}_{\hat{y}_{k+1}} e_k^T$.

To solve linear equations with Lanczos one makes the *ansatz* $x_k = x_0 + Y_k y$. Then

$$Cx_k = Cx_0 + CY_k y = Cx_0 + Y_k T_k y + \hat{y}_{k+1} e_k^T y = b$$

$$\iff Y_k T_k y + \hat{y}_{k+1} e_k^T y = r_0.$$

Multiplying from the left with Z_k^T gives

$$\underbrace{Z_k^T Y_k}_{I} T_k y + \underbrace{Z_k^T \hat{y}_{k+1}}_{0} e_k^T y = Z_k^T r_0.$$

Therefore by choosing $y_0 = r_0/||r_0||$ and putting $\beta = ||r_0||$ one has to compute y as the solution of

$$T_k y = \beta e_1.$$

Note that for the residual $r_k = b - Cx_k$ we have by construction $r_k \perp \mathcal{K}_k(C^T, z_0)$ and furthermore

$$r_k = b - Cx_0 - CY_k y = r_0 - \underbrace{Y_k T_k y}_{r_0} - \hat{y}_{k+1} e_k^T y = -\hat{y}_{k+1} e_k^T y \in \mathcal{K}_{k+1}(C, r_0).$$

For the residual in TEA we have for $m = 0$

$$r(s_{0k}) \in \mathcal{K}_{k+1}(A, r_0) = \mathcal{K}_k(C, r_0).$$

From the normal equations (with Z defined by (30)) we conclude

$$Z^T r(s_{0k}) = 0 \iff r(s_{0k}) \perp \mathcal{K}_k(A^T, q) = \mathcal{K}_k(C^T, q).$$

Therefore again

Theorem 3 *The residuals of Lanczos and TEA lie for $q = z_0$ in the same subspaces and so both algorithms are equivalent.*

An important difference of both algorithms for practical computations is the fact, that the Lanczos algorithm needs both operators for A and A^T while it is sufficient for TEA to have A.

7 Determinantal representation and Shanks transformation

All four extrapolation methods (MPE,RRE,MMPE,TEA) have as we have seen a similar structure. The extrapolated value is expressed as a linear combination of $k + 1$ successive iterates

$$s_{mk} = \sum_{j=0}^{k} \gamma_j x_{m+j},$$

where the coefficients γ_j are determined as solution of the linear system

$$\begin{array}{ccccccccc} \gamma_0 & + & \gamma_1 & + & \cdots & + & \gamma_k & = & 1 \\ u_{00}\gamma_0 & + & \mu_{01}\gamma_1 & + & \cdots & + & \mu_{0k}\gamma_k & = & 0 \\ \vdots & & \vdots & & \cdots & & \vdots & & \vdots \\ \mu_{k-1,0}\gamma_0 & + & \mu_{k-1,1}\gamma_1 & + & \cdots & + & \mu_{k-1,k}\gamma_k & = & 0 \end{array}. \tag{31}$$

The coefficients μ_{ij} of these "normal equations" are

$$\begin{array}{lll} \mu_{ij} & = & (u_{m+i}, u_{m+j}) \quad \text{for MPE;} \\ \mu_{ij} & = & (w_{m+i}, u_{m+j}) \quad \text{for RRE;} \\ \mu_{ij} & = & (q_i, u_{m+j}) \quad \text{for MMPE;} \\ \mu_{ij} & = & (q, u_{m+j+i}) \quad \text{for TEA.} \end{array}$$

Let R denote the matrix of (31). If R_j is R with its j-th column replaced by the right hand side e_1 then using Cramer's rule we have

$$\gamma_j = \frac{\det R_j}{\det R}$$

and therefore

$$s_{mk} = \sum_{j=0}^{k} \gamma_j x_{m+j} = \frac{1}{\det R} \sum_{j=0}^{k} (\det R_j) x_{m+j}. \tag{32}$$

If we now introduce the determinants,

$$D(\sigma_0, \ldots, \sigma_k) := \begin{vmatrix} \sigma_0, & \cdots & \sigma_k \\ \mu_{00}, & \cdots & \mu_{0k} \\ \vdots & \cdots & \vdots \\ \mu_{k-1,0}, & \cdots & \mu_{k-1,k} \end{vmatrix} \tag{33}$$

then *formally* the last sum in (32) can be written as the Laplace expansion of the first row of the determinant $D(x_m, \ldots, x_{m+k})$. Thus we obtain the following expression for s_{mk}, which is a *generalized Shank transformation*

$$s_{mk} = \frac{D(x_m, \ldots, x_{m+k})}{D(1, \ldots, 1)}. \tag{34}$$

8 The scalar ε-algorithm (SEA)

Let $\{x_i\}$ now be a *scalar* sequence with $x_i \to s$ as $i \to \infty$ and with for which the error has an expansion like (4) or more generally

$$\sum_{i=0}^{k} c_i(x_{m+i} - s) = 0, \quad \text{for all} \quad m > N \quad \text{and} \quad \sum_{i=0}^{k} c_i \neq 0.$$

If we consider these equations for $m, m+1, \ldots, m+k$ then for the existence of non trivial solutions the determinant of the matrix has to vanish. Introducing the *Hankel determinant*

$$H_{k+1}(x_m) := \begin{vmatrix} x_m & x_{m+1} & \cdots & x_{m+k} \\ x_{m+1} & x_{m+2} & \cdots & x_{m+k+1} \\ \vdots & \vdots & \cdots & \vdots \\ x_{m+k} & x_{m+k+1} & \cdots & x_{m+2k} \end{vmatrix}$$

it is shown in [1], using some manipulations with determinants, that

$$s = \frac{H_{k+1}(x_m)}{H_k(\Delta^2 x_m)}. \tag{35}$$

Since k is generally unknown, the right hand side of (35) is considered for various values of m and k and is called the *Shanks transformation* of the sequence x_j. It is often denoted by s_{mk} or $e_k(x_m)$.

Wynn discovered a simple recursion *that allows the computation of $e_k(x_m)$ without solving linear systems or computing determinants: the ε-algorithm.*

Let $\varepsilon_{-1}^{(n)} = 0$ and $\varepsilon_0^{(n)} = x_n$ for $n = 0, 1, 2, \ldots$. From these values, the following table using the recurrence relation

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}} \tag{36}$$

is constructed:

$$\begin{array}{ccccccc}
\varepsilon_{-1}^{(0)} & & & & & & \\
 & \varepsilon_0^{(0)} & & & & & \\
\varepsilon_{-1}^{(1)} & & \varepsilon_1^{(0)} & & & & \\
 & \varepsilon_0^{(1)} & & \varepsilon_2^{(0)} & & & \\
\varepsilon_{-1}^{(2)} & & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} & & \\
 & \varepsilon_0^{(2)} & & \varepsilon_2^{(1)} & & \cdots & \\
\varepsilon_{-1}^{(3)} & & \varepsilon_1^{(2)} & & \cdots & & \\
 & \varepsilon_0^{(3)} & & \cdots & & &
\end{array} \tag{37}$$

The theorem of Wynn then states that

$$\varepsilon_{2k}^{(n)} = e_k(x_n) \quad \text{and} \quad \varepsilon_{2k+1}^{(n)} = \frac{1}{e_k(\Delta x_n)}.$$

For a vector sequence we can apply the ε-algorithms to each component. We call the resulting algorithm SEA. The main disadvantage of SEA is, *that it 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 exists or be quite large numerically* [15]. In [1], modified recursions are mentioned that eliminates that problem.

9 The vector ε-algorithms VEA and TEA

In order to generalize the ε-scheme for a vector sequence $\{x_j\}$ we have to give a meaning to the "inverse of a vector" used in the recursion (36). One possibility is to use the pseudoinverse (Samelson inverse), defined by

$$y^{-1} := \frac{1}{\|y\|^2}\bar{y}^T. \tag{38}$$

It has been conjectured by Wynn and proved by McLeod that if the sequence $\{x_j\}$ has the property that

$$\sum_{i=0}^{k} c_i(x_{m+i} - s) = 0, \quad m > N$$

then $\varepsilon_{2k}^{(m)} = s$. This is however all that ones knows about VEA. No determinantal representations of $\varepsilon_i^{(n)}$ are known.

A second generalization for vector sequences is based on a different interpretation of "inverse" [1]. Brezinski defines the inverse of a ordered pair (a, b) of vectors such that $a^T b \neq 0$ to be the ordered pair (b^{-1}, a^{-1}), where

$$b^{-1} = \frac{a}{b^T a}, \quad a^{-1} = \frac{b}{b^T a}.$$

For real sequences $\{x_j\}$ two algorithms result with this interpretation [17], where the first refers to TEA defined in section 6 (note that the difference in $\Delta\varepsilon_m^{(n)}$ is with respect to n).

Choose an arbitrary vector q and set

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

Algorithm 1: (TEA1)

$$\varepsilon_{2m+1}^{(n)} = \varepsilon_{2m-1}^{(n+1)} + \frac{q}{q^T \Delta\varepsilon_{2m}^{(n)}}, \varepsilon_{2m+2}^{(n)} = \varepsilon_{2m}^{(n+1)} + \frac{\Delta\varepsilon_{2m}^{(n)}}{\Delta\varepsilon_{2m+1}^{(n)}{}^T \Delta\varepsilon_{2m}^{(n)}} \qquad m, n = 0, 1 \ldots \tag{39}$$

Algorithm 2: (TEA2)

$$\varepsilon_{2m+1}^{(n)} = \varepsilon_{2m-1}^{(n+1)} + \frac{q}{q^T \Delta\varepsilon_{2m}^{(n)}}, \varepsilon_{2m+2}^{(n)} = \varepsilon_{2m}^{(n+1)} + \frac{\Delta\varepsilon_{2m}^{(n+1)}}{\Delta\varepsilon_{2m+1}^{(n)}{}^T \Delta\varepsilon_{2m}^{(n+1)}} \qquad m, n = 0, 1 \ldots \tag{40}$$

As it is shown in [1] TEA1 computes

$$\varepsilon_{2k}^{(n)} = e_k(x_n)$$

and yields the same extrapolated values as if it is computed using the "normal equations" (31). It is possible that the recursive computation will be numerically more stable, since then the coefficients of the minimal polynomial are not computed.

10 Numerical experiments

In this section we report a number of numerical experiments executed on a CRAY X-MP/2 using 64-bit real arithmetic. We have compared the Lanczos and different ε-algorithms using the following Model Problem (found in [8]):

We solve the elliptic partial differential equation in the region $\Omega = (0,1) \times (0,1)$

$$-\Delta u + \gamma(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y}) + \beta u = g \qquad \text{in } \Omega$$

with Dirichlet boundary conditions on $\partial\Omega$. This problem is discretized using centered differences for both the first and second order derivatives on a uniform mesh with red-black ordering. The mesh size $h = 1/32$ leads to 961 unknowns. The boundary conditions (i.e. the right hand side vector) is chosen such that the solution u to the *discrete* system is one everywhere. This allows an easy verification of the result. The nonsymmetrical linear system has now the form $G(\gamma, \beta)u = f$.

The parameters γ and β can be varied to make the problem more or less difficult to solve. The greater γ is chosen, the more unsymmetrical the system becomes, and the value of β makes the system more or less positive definite. Let's define $\delta = \gamma h/2$.

The basic linear stationary iterative method ($x_{k+1} = Ax_k + b$) on which we extrapolate the solution is either Point-Jacobi or Gauss-Seidel. The corresponding subspace method then solves the related linear system ($Cx = b$ with $C = I - A$). In order to avoid operating in a special subspace, we choose the initial approximation x_0 as a normed random vector.

In our Figures, we show the behaviour of the infinity norm of the *error*-vector over the number of iteration steps or over the elapsed time on our CRAY.

First, we want to show numerically the equivalence between the non-symmetric Lanczos, BiCG and the TEA1. Therefore we choose $\gamma = 96$ ($\delta = 1.5$) and $\beta = 0$. In Fig 1. we see, that the three methods are

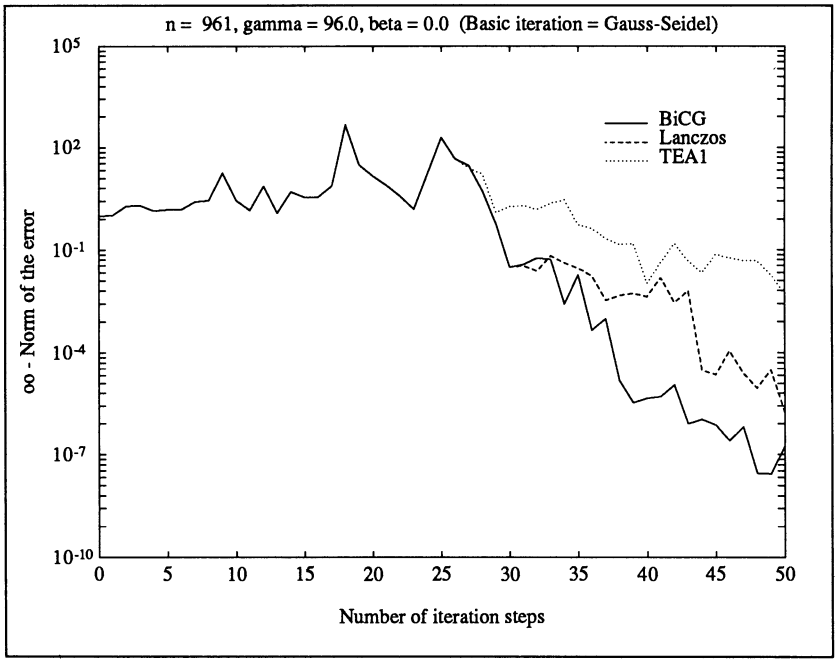

Figure 1: Equivalence of Lanczos and TEA1

equivalent as mentioned in Theorem 3. (Note that one ε-Step uses *two* basic iteration steps.) Lanczos and TEA1 branch off due to rounding errors after 27 steps, Lanczos and BiCG after 30.

However this is not a good way to use the ε-algorithm, because the deeper the ε-scheme is computed the more time is spent, the more memory is used and the more rounding errors influence the results. In finite arithmetic one has to stop the computation of every ε-scheme after some rows and columns. With the best approximate of this scheme, one starts a new ε-scheme. We can describe this procedure in the following algorithm:

1. Choose x_0 and numbers n_k and m_k.

2. for $k = 0, 1, \ldots$ do

$$\left.\begin{array}{lcll} y_0 & = & x_k & \\ y_{i+1} & = & Ay_i, & i = 0, \ldots, n_k - 1 \end{array}\right\} \text{ Basic Iteration}$$

$$\left.\begin{array}{lcll} \varepsilon_0^{(0)} & = & y_{n_k} & \\ \varepsilon_0^{(i+1)} & = & A\varepsilon_0^{(i)}, & i = 0, \ldots, 2m_k - 1 \\ \varepsilon_{2m_k}^{(0)} & = & e_{m_k}(y_{n_k}) & \end{array}\right\} \text{ Extrapolation}$$

$$x_{k+1} = \varepsilon_{2m_k}^{(0)} \quad \} \text{ best approximate}$$

In Fig 2, we have applied this truncated extrapolation to the same problem as in Fig 1. We first performed $n_0 = 35$ (low cost) Gauss-Seidel steps and started then the extrapolation of width $m_0 = 16$,

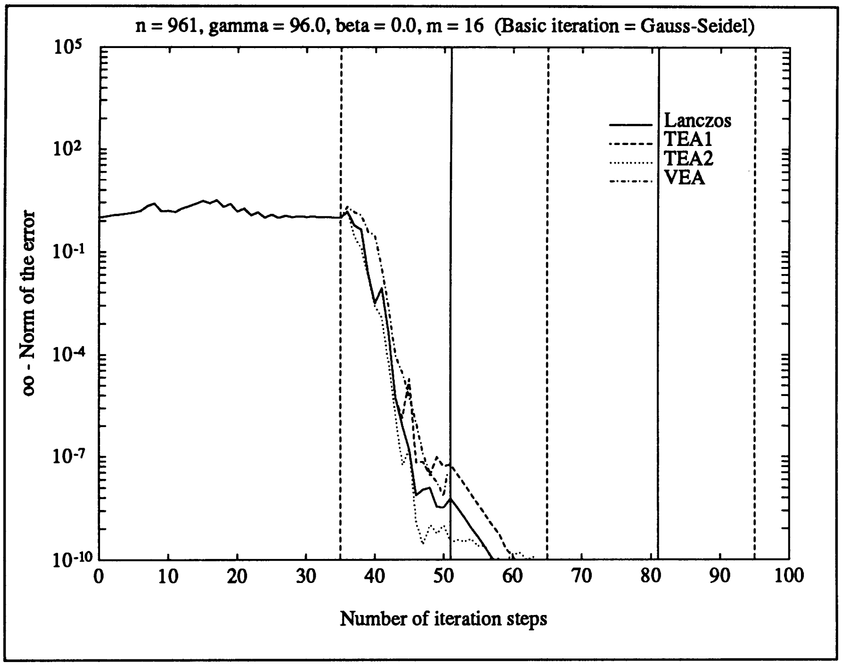

Figure 2: $\delta = 1.5, n_0 = 35$

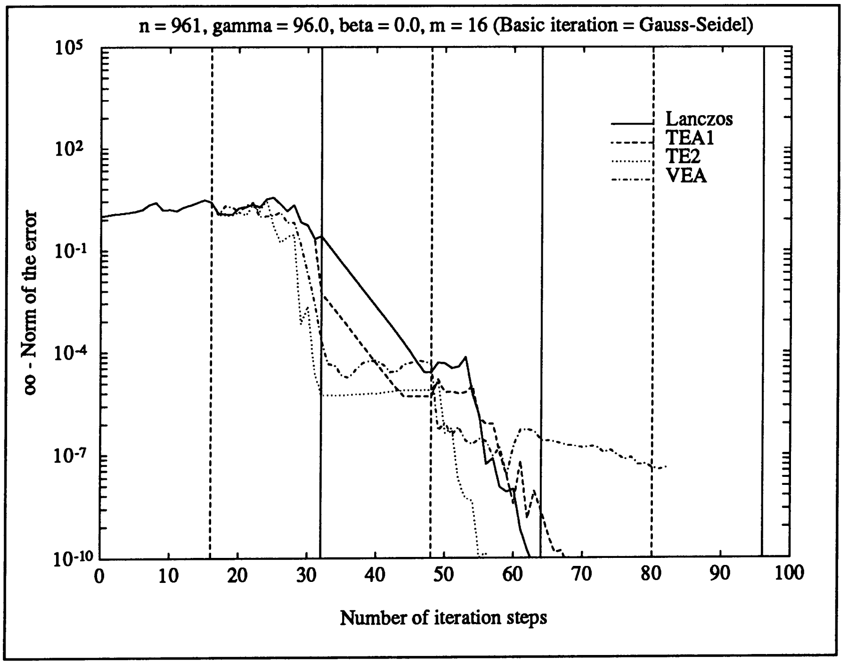

Figure 3: $\delta = 1.5, n_0 = 16$

i.e. we had to compute another 32 Gauss-Seidel steps as basis for the ε-scheme (37). Then we again compute simple basic iteration steps starting with the best approximate ($\varepsilon_{16}^{(0)}$ in our example)., which made the error drop rapidly below 10^{-10}. (In the Figures, these different parts are separated by vertical lines: basic iteration steps between a solid and a dashed line and extrapolation steps between a dashed and a solid line.) We have compared Lanczos, TEA1, TEA2 and VEA, which breaks down after 48 iteration steps.

We see, that we gain up to 10 digits during the extrapolation phase. Looking at TEA1 and Lanczos we see, that after the extrapolation steps, the contributions z_l of the largest eigenvalues of the iteration matrix to the error (cf (3)) must have disappeared in some sense and therefore we get a good convergence rate.

In Fig 2, we see also, that the ε-Algorithm is much more powerful, if we first compute *many* Gauss-Seidel steps. In Fig 3 however, n_0 is set only to 16, and we see, that we use *two* extrapolation phases of width 16 ($m_k = 16, k \geq 0$) to reach the same accuracy as in Fig 2. We can also see, that we gain in each of these two phases only about 5 digits due to the smaller n_0.

In Fig 3 we also notice the difference between the TEA1 and the other two extrapolation methods, TEA2 and VEA. While by TEA1 the contributions of the largest eigenvalues of the iteration matrix disappeared after the extrapolation scheme, we cannot observe this effect by TEA2 and VEA, but the two latter methods gain more accuracy during their extrapolation phase.

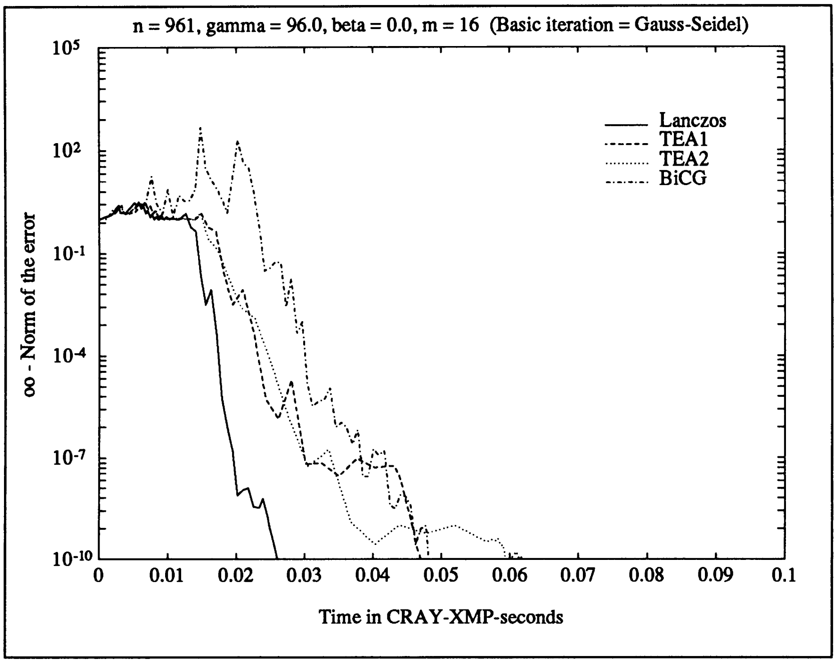

Figure 4: $\delta = 2, n_0 = 35$, related to time

Looking at Fig 4, where the same computation is shown as in Fig 2 but related to time, we see, that all four compared algorithms (Lanczos, TEA1, TEA2 and BiCG) use about the same time (Lanczos is fastest in this example). But the big advantage of the ε-algorithms is, that they don't need the transposed operator of the system!

In the next example (see Fig. 4), we want to show, that the extrapolation works, even if the basic iteration is not convergent. For that, we have chosen with $\gamma = 128$ ($\delta = 2$) and $\beta = 0$ a more unsymmetric example. We are again using Gauss-Seidel as the basic iteration. The width of the extrapolation is always $m = 14$ (based on 28 basic iteration steps) and we always compute $n_k = 23$ intermediate Gauss-Seidel steps. We again see, that after the extrapolation steps of TEA1 or Lanczos, the convergence rate of

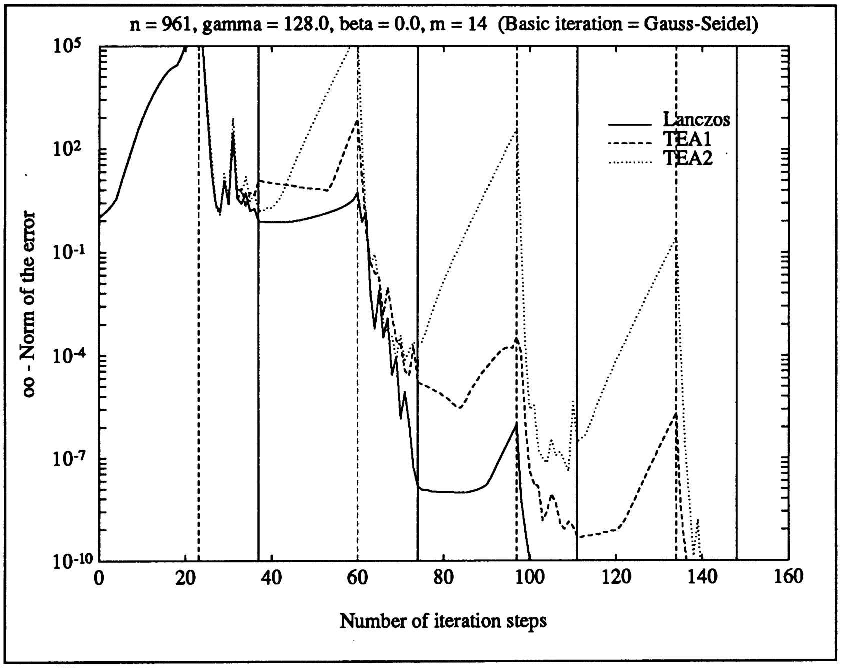

Figure 5: $\delta = 2$

the basic iteration is better for some steps, until the contribution of the largest eigenvalue to the error reappears.

We have observed, that the more Gauss-Seidel steps are performed, the more effective the extrapolation works. Without Gauss-Seidel steps, we would use in this example at least 30 extrapolation steps, that means an extra storage of 60 vectors. This storage is also used when solving the problem with GCR(30).

Conclusion

- We have seen, that the deeper the ε-scheme is computed, the more cancellation occurs. Therefore only some rows and columns of the ε-scheme contain satisfactory approximations. That requires finding stable methods to compute this scheme (like BiCG, which seems to be a good algorithm for TEA2), or one has to transform the formulas in numerically favorable forms. (e.g. by extrapolating the differences u_i of the basic iteration x_i instead of the basic iteration itself.)

- The ε-algorithm is motivation to split the computation into two parts and to use a truncated extrapolation. The BiCG and Lanczos methods don't emphasis this point so this is an interesting new aspect.

- The main problem in the truncated extrapolation is to find the best values for n_k and m_k in every step, preferably in an adaptive way. The values of n_k and m_k can easily be changed during the computation, because the ε-scheme is (favorably) computed row wise. This tuning of n_k and m_k needs more research.

- It seems to us, that the extrapolation works very well even if the system is singular. Such a problem has been discussed in [4] by using the ε-algorithm, but in this area there are still many open questions.

References

[1] C. Brezinski: *Accélération de la Convergence en Analyse Numérique,* Lecture notes in Mathematics 584, Springer 1977

[2] A. Buja, Hastie and Tibshirani: *Linear Smoothers and Additive Models,* Annals of Statistics, Vol. 17, No. 2, June 1989.

[3] J.P. Delahaye: *Sequence Transformations,* Springer 1988

[4] W. Gander and G. Golub: *Discussion of Buja A., Hastie and Tibshirani: Linear Smoothers and Additive Models,* Annals of Statistics, Vol. 17, No. 2, June 1989

[5] W.F. Ford and A. Sidi: *Recursive Algorithms for Vector Extrapolation Methods,* Applied Numerical Mathematics 4, (1988), p. 477-489.

[6] H. Ortloff: *Asymptotic Behaviour and Acceleration of Iterative Sequences,* Numer. Math. 49, (1986), p. 545-559.

[7] Y. Saad: *The Lanczos Biorthogonalization Algorithm and other oblique projection methods for solving large unsymmetric systems,* SIAM J. Numeri. Anal., Vol. 19, No. 3, June 1982.

[8] Y. Saad: *Preconditioning techniques for nonsymmetric and indefinite linear systems,* Journal of Computational and Applied Mathematics 24, (1988), p 89-105.

[9] A. Sidi: *Convergence and Stability Properties of Minimal Polynomial and Reduced Rank Extrapolation Algorithms,* SIAM J. Numeri. Anal., Vol. 23, No.1, February 1986, p. 178-196.

[10] A. Sidi and J. Bridger: *Convergence and stability Analysis for some Vector Extrapolation Methods in the Presence of Defective Iteration Matrices,* Journal of Computational and Applied Mathematics 22, (1988), p. 35-61.

[11] A. Sidi: *Extrapolation vs. Projection methods for linear Systems of Equations,* Journal of Computational and Applied Mathematics 22, (1988), p. 77-88.

[12] A. Sidi: *Application of Vector Extrapolation Methods to Consistent Singular Linear Systems,* Technical Report # 540 Technion, February 1989.

[13] A. Sidi and M.L. Celestina: *Convergence Acceleration for Vector Sequences and Applications to Computational Fluid Dynamics,* NASA Technical Memorandum 101327, ICOMP-88-17, 1988

[14] A. Sidi, W.F. Ford and D.A. Smith, *Acceleration of Convergence of Vector Sequences,* SIAM J. Numeri. Anal., Vol. 23, No.1, February 1986, p. 178-196.

[15] D.A. Smith, W.F. Ford and A. Sidi: *Extrapolation Methods for Vector Sequences,* SIAM Review, Vol. 29, No.2, June 1987, p. 199-233.

[16] D.A. Smith, W.F. Ford and A. Sidi: *Correction to "Extrapolation Methods for Vector Sequences",* SIAM Review, Vol. 30, No. 4, December 1988

[17] R.C.E. Tan: *Implementation of the topological ε-Algorithm,* SIAM J. Sci. Stat. Comput. Vol. 9, No. 5. September 1988

[18] R.S. Varga: *Matrix Iterative Analysis,* Prentice-Hall Inc., New Jersey, 1962

[19] J.H. Wilkinson: *The Algebraic Eigenvalue Problem,* Claredon Press, Oxford, 1988