Numer. Math. 35, 175–187 (1980)

Numerische Mathematik

A General Extrapolation Algorithm

C. Brezinski

UER IEEA – informatique, Université de Lille 1, 59655 Villeneuve d'Ascq cédex, France

Summary. In this paper a general formalism for linear and rational extrapolation processes is developped. This formalism includes most of the sequence transformations actually used for convergence acceleration. A general recursive algorithm for implementing the method is given. Convergence results and convergence acceleration results are proved. The vector case and some other extensions are also studied.

Subject Classification: AMS(MOS): 65B, 65B05, CR: 5.13, 5.19.

Extrapolation processes are very useful in numerical analysis since many methods produce sequences of approximations $\{S_n\}$ (often depending on a sequence of parameters $\{x_n\}$) of the exact result S. If the error has the form:

$$S_n - S = a_1 g_1(n) + a_2 g_2(n) + \ldots$$

where the g_i are known sequences, then more accurate approximations to S can be obtained by using an extrapolation method. A typical example of this situation is given by the well known trapezoidal rule for quadratures whose results can be extrapolated by Romberg method.

The aim of this paper is to provide a general formalism to study such extrapolation methods in the general case. This formalism includes most of the sequence transformations actually used for convergence acceleration. A recursive algorithm for implementing this general extrapolation method is given and convergence theorems are proved. The vector case is also studied and some extensions are developed.

1. Linear Extrapolation Processes

The basic idea of linear extrapolation processes is to assume that the scalar sequence $\{S_n\}$ to be extrapolated behaves like

$$S_n = S + a_1 g_1(n) + \ldots + a_k g_k(n) \tag{1}$$

0029-599X/80/0035/0175/$02.60

where the sequences g_i are given sequences and to take S as the extrapolated value of $\{S_n\}$. In general S is an approximate value of the limit of the sequence $\{S_n\}$ if it converges or an approximate value of its antilimit when it diverges.

S is computed by solving the linear system

$$S_{n+i}=S+a_1 g_1(n+i)+\ldots+a_k g_k(n+i) \qquad i=0,\ldots,k. \tag{2}$$

We shall always assume in this paper that such systems are nonsingular.

If the sequence $\{S_n\}$ has not the exact form (1), then the value of S obtained by solving the preceding system will depend on the indexes n and k. We shall denote it by $E_k(S_n)$ and, from (2), it is easy to see that (compare with [8, p 40])

$$E_k(S_n)=\frac{\begin{vmatrix} S_n & S_{n+1} & \ldots & S_{n+k} \\ g_1(n) & g_1(n+1) & \ldots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \ldots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \ldots & 1 \\ g_1(n) & g_1(n+1) & \ldots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \ldots & g_k(n+k) \end{vmatrix}}. \tag{3}$$

This formalism is quite general since it includes many very well known sequence transformations which are used to accelerate the convergence of the original sequence $\{S_n\}$ [3]:

- $g_i(n)=x_n^i$ — Richardson extrapolation process ($\{x_n\}$ is a given auxiliary sequence of parameters).
- $g_i(n)=x_{n+i-1}$ — G-transformation [9].
- $g_i(n)=x_i^n$ — Summation processes.
- $g_i(n)=\Delta S_{n+i-1}$ or $g_i(n)=\Delta^i S_n$ — Shanks transformation [16].
- $g_i(n)=(\Delta S_n)^i$ — Germain-Bonne transformation [8].
- $g_i(n)=x_n^{i-1}\,\Delta S_n/g(n)$ — Levin generalized transformation [13] (g is a given sequence).
- $g_i(n)=R^i(S_n\,\Delta x_n)/\Delta x_n$ — First generalization of the ε-algorithm [3] where R is an operator which generalizes Δ.
- $g_1(n)=x_n$, $g_i(n)=\Delta S_{n+i-2}$ — Process p [3].

This formalism also includes rational extrapolation of the form:

$$S_n=\frac{S+b_1 f_1(n)+\ldots+b_m f_m(n)}{1+c_1 h_1(n)+\ldots+c_p h_p(n)}$$

since we have:

$$S_n=S+b_1 f_1(n)+\ldots+b_m f_m(n)-c_1 S_n h_1(n)-\ldots-c_p S_n h_p(n)$$

which can be written as (1) by setting:

$$g_1(n)=f_1(n),\ \ldots,\ g_m(n)=f_m(n), \qquad g_{m+1}(n)=S_n h_1(n),\ \ldots,\ g_{m+p}(n)=S_n h_p(n).$$

As a particular case of rational extrapolation let us mention extrapolation by reciprocal differences which is obtained by taking $f_i(n)=h_i(n)=x_n^{-i}$ for $i=1, \ldots, k/2$ [20].

If the sequences g_i do not depend on $\{S_n\}$ we have a linear sequence transformation while the process is nonlinear if the g_i depend on $\{S_n\}$.

2. A Recursive Algorithm

The main algorithmic problem is now to compute recursively the numbers $E_k(S_n)$ without computing the determinants appearing in (3) since a numerical analyst does not know how to compute determinants.

Such a recursive algorithm exists for each of the particular cases of sequence transformations mentioned above, the most well known of which is perhaps Wynn ε-algorithm [19] for implementing Shanks transformation. Richardson process and Germain-Bonne transformation can be implemented by Neville-Aitken scheme, the G-transformation by the G-algorithm [15], Levin transformation by some recursive methods [6], Shanks transformation by the ε-algorithm or by some other devices [5] and so on. Fortran subroutines corresponding to these algorithms can be found in [4].

The major trouble of such a situation is that one has to use a different algorithm in each particular case and that, up to now, a new algorithm had to be exhibited for each new choice of the sequences g_i.

In this section we shall present a general recursive algorithm for computing the numbers $E_k(S_n)$ for all k and n which works whatever be the g_i's. Of course for some particular sequence transformations a well adapted algorithm could be more powerful (with respect to the number of arithmetic operations) than this general one.

This algorithm, which will be called the E-algorithm, is the following:

initializations $E_0^{(n)}=S_n \qquad n=0, 1, \ldots$

$g_{0,i}^{(n)}=g_i(n) \qquad i=1, 2, \ldots$ and $n=0, 1, \ldots.$

For $k=1, 2, \ldots$ and $n=0, 1, \ldots$

$$E_k^{(n)}=\frac{E_{k-1}^{(n)} g_{k-1,k}^{(n+1)}-E_{k-1}^{(n+1)} g_{k-1,k}^{(n)}}{g_{k-1,k}^{(n+1)}-g_{k-1,k}^{(n)}}$$

$$g_{k,i}^{(n)}=\frac{g_{k-1,i}^{(n)} g_{k-1,k}^{(n+1)}-g_{k-1,i}^{(n+1)} g_{k-1,k}^{(n)}}{g_{k-1,k}^{(n+1)}-g_{k-1,k}^{(n)}} \qquad i=k+1, k+2, \ldots.$$

Theorem 1.

$$E_k^{(n)}=E_k(S_n) \qquad k, n=0, 1, \ldots.$$

Proof. By induction. We shall make use of an abreviate notation where a determinant is only denoted by the elements of its first row or column. Thus

$$E_k(S_n)=\frac{|S_n g_1(n) \ldots g_k(n)|}{|1 g_1(n) \ldots g_k(n)|}.$$

We shall prove simultaneously that

$$g_{k,i}^{(n)}=\frac{|g_i(n)\, g_1(n)\dots g_k(n)|}{|1\, g_1(n)\dots g_k(n)|}.$$

The property is obviously true for $k=0$. Let us assume that it is true up to the index $k-1$. Then replacing the quantities by their expressions in the rule of the algorithm and reducing to the same denominator we get:

$$E_k^{(n)}=N_k^{(n)}/D_k^{(n)}$$

with

$$\begin{aligned}N_k^{(n)}=&|S_n\, g_1(n)\dots g_{k-1}(n)|\,|g_k(n+1)\, g_1(n+1)\dots g_{k-1}(n+1)|\\&-|S_{n+1}\, g_1(n+1)\dots g_{k-1}(n+1)|\,|g_k(n)\, g_1(n)\dots g_{k-1}(n)|\end{aligned}$$

and with

$$\begin{aligned}D_k^{(n)}=&|1\, g_1(n)\dots g_{k-1}(n)|\,|g_k(n+1)\, g_1(n+1)\dots g_{k-1}(n+1)|\\&-|1\, g_1(n+1)\dots g_{k-1}(n+1)|\,|g_k(n)\, g_1(n)\dots g_{k-1}(n)|\end{aligned}$$

and an analoguous relation for $g_{k,i}^{(n)}$ by replacing S_n by $g_i(n)$ in the determinants.

By Sylvester determinantal identity [7] we have:

$$\begin{aligned}&|g_1(n+1)\dots g_{k-1}(n+1)|\,|S_n\, g_1(n)\dots g_k(n)|\\&\quad=|S_n\, g_1(n)\dots g_{k-1}(n)|\,|g_1(n+1)\dots g_k(n+1)|\\&\qquad-|S_{n+1}\, g_1(n+1)\dots g_{k-1}(n+1)|\,|g_1(n)\dots g_k(n)|\end{aligned}$$

which is the numerator of the preceding expression multiplied by $(-1)^{k-1}$. Sylvester identity also applies to the denominator of $E_k^{(n)}$ and to $g_{k,i}^{(n)}$ and the result is proved. ■

The E-algorithm is obviously a generalization of Richardson extrapolation algorithm which is based on Neville-Aitken scheme for computing interpolation polynomials. The E-algorithm can be also derived from a generalization of Neville-Aitken scheme due to Mühlbach [14] for interpolation by a Chebyshev system. For another generalization see [17].

The quantities $E_k^{(n)}$ are usually placed in a double array, the E-array, where the lower index denotes a column and the upper index a descending diagonal as follows:

$$\begin{array}{llll}E_0(S_0)=S_0 & & & \\ E_0(S_1)=S_1 & E_1(S_0) & & \\ E_0(S_2)=S_2 & E_1(S_1) & E_2(S_0) & \\ E_0(S_3)=S_3 & E_1(S_2) & E_2(S_1) & E_3(S_0)\\ \vdots & \vdots & \vdots & \vdots\ \ddots\end{array}$$

The $g_{k,i}^{(n)}$ can also be placed in similar arrays for a fixed value of i. Such arrays will be called the g_i-arrays for $i=1,2,\dots$. From the determinantal expression of $g_{k,i}^{(n)}$ we see that $g_{k,i}^{(n)}=0$ for $k\geqq i$; thus the g_i-array only contains the columns 0 up to $i-1$.

Some other determinantal identities can be found between the members of the E-array. For example if each column in (3) is replaced by its difference with the preceding one, then:

$$E_k(S_n)=\frac{\begin{vmatrix} S_n & g_1(n) & \dots & g_k(n) \\ \Delta S_n & \Delta g_1(n) & \dots & \Delta g_k(n) \\ \hline \Delta S_{n+k-1} & \Delta g_1(n+k-1) & \dots & \Delta g_k(n+k-1) \end{vmatrix}}{|\Delta g_1(n) \dots \Delta g_k(n)|}.$$

Using again Sylvester identity for the numerator of $E_{k+1}(S_n)$ we get:

$$E_{k+1}(S_n)-E_k(S_n)=-\frac{|\Delta S_n\, \Delta g_1(n) \dots \Delta g_k(n)|\, |g_1(n) \dots g_{k+1}(n)|}{|\Delta g_1(n) \dots \Delta g_k(n)|\, |\Delta g_1(n) \dots \Delta g_{k+1}(n)|}.$$

This relation generalizes a similar relation proved by Wynn [23] for Shanks transformation.

We shall now discuss the practical implementation of the E-algorithm on a computer. There are two ways for using such algorithms; the first one is an a posteriori utilization which consists in computing a fixed number k of terms of the sequence $\{S_n\}$ to be extrapolated and then applying the E-algorithm by column. The disadvantage of this use is that all the computations would have to be started again from the beginning if one wants to increase k.

The second possible use of the E-algorithm is a parallel use which consists in constructing the E-array as far as possible after each computation of a new term of the sequence $\{S_n\}$. This use is much more convenient than the first one but the implementation of the algorithm is more complicated since the computation of the E-array must be done line by line. The parallel implementation of the algorithm is as follows:

- initializations: $E_0^{(0)}=S_0 \quad g_{0,1}^{(0)}=g_1(0)$
- $k=1, 2, \dots$.

The following quantities have been stored in the computer's memory during the previous stages of the algorithm:

- line of the E-array: $E_0^{(k-1)}, E_1^{(k-2)}, \dots, E_{k-1}^{(0)}$
- line and last column of the g_i-arrays for $i=1, \dots, k-1$:
$$g_{0,i}^{(k-1)}, g_{1,i}^{(k-2)}, \dots, g_{i-2,i}^{(k-i+1)} \text{ and } g_{i-1,i}^{(0)}, \dots, g_{i-1,i}^{(k-i)}.$$

We shall now enter the values of $g_{0,1}^{(k)}=g_1(k), \dots, g_{0,k-1}^{(k)}=g_{k-1}(k)$ and compute the next line of the g_i-arrays for $i=1, \dots, k-1$ (if $k>2$). For that we compute $g_{j,i}^{(k-j)}$ for $i=2, \dots, k-1$ and $j=1, \dots, i-1$. The new line and the new last column to be stored will be:

$$g_{0,i}^{(k)}, g_{1,i}^{(k-1)}, \dots, g_{i-2,i}^{(k-i+2)} \quad \text{and} \quad g_{i-1,i}^{(0)}, g_{i-1,i}^{(1)}, \dots, g_{i-1,i}^{(k-i+1)}.$$

This can be automatically done by using Wynn's moving lozenge technique for avoiding unuseful storage [22]. We shall now construct the g_k-array by lines. We first have to enter the value of $g_{0,k}^{(0)}=g_k(0)$. Then, for $i=1, \dots, k$, we enter $g_{0,k}^{(i)}$

$=g_k(i)$ and for $j=1, \ldots, i$ ($i-1$ when $i=k$) we compute $g_{j,k}^{(i-j)}$ by the formula of the E-algorithm. The line of the g_k-array contains $g_{0,k}^{(k)}, g_{1,k}^{(k-1)}, \ldots, g_{k-2,k}^{(2)}$ and the last column contains $g_{k-1,k}^{(0)}$ and $g_{k-1,k}^{(1)}$. We are now ready to compute the new line of the E-array. We enter $E_0^{(k)}=S_k$ and for $i=1, \ldots, k$ we compute $E_i^{(k-i)}$.

The computation of $E_k^{(0)}$ by this method requires $k(k+1)(2k+1)/3$ multiplications, the same number of additions and half this number of divisions. That is, for large k, approximatively $5k^3/3$ arithmetic operations.

The fundamental algebraic property of our general extrapolation process is

Theorem 2. *If* $S_n=S+a_1 g_1(n)+\ldots+a_k g_k(n), \forall n$ *then* $E_k(S_n)=S, \forall n$.

More generally we have:

Theorem 3. *If* $S_n=S+a_1 g_1(n)+a_2 g_2(n)+\ldots, \forall n$ *then*

$$E_k(S_n)=S+a_{k+1} g_{k,k+1}^{(n)}+a_{k+2} g_{k,k+2}^{(n)}+\ldots, \forall k, n.$$

Proof. By induction. The property is true for $k=0$. Let us assume that it is true up to the index k and let us replace in the E-algorithm. The property immediately follows from the recurrence relation of $g_{k+1,i}^{(n)}$. Theorem 2 derives also from here since $a_i=0$ for $i>k$. ■

From (3) we see that $E_k^{(n)}$ and $g_{k,i}^{(n)}$ can be written as

$$E_k^{(n)}=\sum_{j=0}^{k} A_j^{(k,n)} S_{n+j}; \qquad g_{k,i}^{(n)}=\sum_{j=0}^{k} A_j^{(k,n)} g_i(n+j)$$

with $\sum_{j=0}^{k} A_j^{(k,n)}=1$.

Replacing in the E-algorithm we see that:

$$A_0^{(k,n)} \Delta g_{k-1,k}^{(n)}=A_0^{(k-1,n)} g_{k-1,k}^{(n+1)}$$

$$A_j^{(k,n)} \Delta g_{k-1,k}^{(n)}=A_j^{(k-1,n)} g_{k-1,k}^{(n+1)}-A_{j-1}^{(k-1,n+1)} g_{k-1,k}^{(n)} \qquad j=1, \ldots, k-1$$

$$A_k^{(k,n)} \Delta g_{k-1,k}^{(n)}=-A_{k-1}^{(k-1,n+1)} g_{k-1,k}^{(n)}$$

with

$$A_0^{(0,n)}=1 \ \forall n \quad \text{and} \quad \Delta g_{k-1,k}^{(n)}=g_{k-1,k}^{(n+1)}-g_{k-1,k}^{(n)}.$$

These relations can be used for the parallel implementation of the extrapolation process instead of using the E-algorithm. This method is particularly well adapted to extrapolate simultaneously several sequences with the same family of sequences g_i.

After this paper was sent for publication I found a paper by Håvie [11] also dealing with general extrapolation schemes. Håvie assumes the existence of the expansion

$$T(h)=T(0)+\sum_{i=1}^{k} a_i e_i(h)+R_k(h)$$

where e_j are known functions satisfying $\lim_{h\to 0} e_{i+1}(h)/e_i(h)=0$ and $R_k(h)=O(e_{k+1}(h))$. An approximate value $T_k^{(n)}$ of $T(0)$ is obtained by solving the system

$$T(h_{n+i})=T_k^{(0)}+\sum_{j=1}^{k} a_j' e_j(h_{n+i}) \qquad i=0,\ldots,k$$

where $h_0>h_1>\ldots>0$. Håvie derives a recursive algorithm to obtain the $T_k^{(n)}$ which is exactly the same as the algorithm described in this section. He then studies a generalization of his scheme, some algebraic and stability properties and particular choices of the e_i and the h_n related to quadrature methods with weight functions singularities.

Since the motivations in Håvie's work were not the same as above the two approaches differ in some points. Håvies does not discuss at all convergence acceleration methods; thus he does not deal with determinantal expressions for $E_k(S_n)$ and $g_{k,i}^{(n)}$ and his proof for the recursive scheme is quite different. It must be noticed that the assumptions on the functions e_i and on the step sequence $\{h_n\}$ are not needed for deriving the algorithm. Theorems 2 and 3 above also follow from Håvie's work as well as the computation of the coefficients $A_i^{(k,n)}$. Håvie does not give convergence and convergence acceleration results and he does not study the vector case. Thus the two papers can be considered as complementary.

3. Convergence Results

Let us now assume that the sequence $\{S_n\}$ converges to S. We shall study the convergence of the columns of the E-array to S. The convergence of the diagonals has not yet been attacked since it is a much more difficult problem. We shall also study convergence acceleration properties of the columns.

Theorem 4. *If* $\lim_{n\to\infty} E_{k-1}(S_n)=S$ *and if* α *and* β *exist such that* $0<\alpha<1<\beta$ *and* $g_{k-1,k}^{(n+1)}/g_{k-1,k}^{(n)}\notin[\alpha,\beta]\ \forall n>N$ *then* $\lim_{n\to\infty} E_k(S_n)=S$.

Proof. It is a consequence of Toeplitz theorem. The direct proof from the E-algorithm is also obvious. ■

This theorem is not very interesting since it is a difficult matter to know if the condition is satisfied or not. We shall now give a condition on the sequences g_i such that the preceding condition be satisfied.

Theorem 5. *If* $\lim_{n\to\infty} S_n=S$, *if* $\lim_{n\to\infty} g_i(n+1)/g_i(n)=b_i\neq 1\ \forall i$, *if* $b_i\neq b_j\ \forall i\neq j$ *then* $\lim_{n\to\infty} E_k(S_n)=S\ \forall k$.

Proof. We shall prove that, under the assumptions of the theorem, $\lim_{n\to\infty} g_{k,i}^{(n+1)}/g_{k,i}^{(n)}=b_i\ \forall i$ and k, which, by Theorem 4, will imply the convergence to S of all the columns of the E-array. The proof is by induction. If the property is assumed to

be true up to the index $k-1$ then:

$$\frac{g_{k,i}^{(n+1)}}{g_{k,i}^{(n)}}=\frac{g_{k-1,k}^{(n+2)}/g_{k-1,k}^{(n+1)}-g_{k-1,i}^{(n+2)}/g_{k-1,i}^{(n+1)}}{g_{k-1,k}^{(n+1)}/g_{k-1,k}^{(n)}-g_{k-1,i}^{(n+1)}/g_{k-1,i}^{(n)}}\cdot\frac{g_{k-1,k}^{(n+1)}/g_{k-1,k}^{(n)}-1}{g_{k-1,k}^{(n+2)}/g_{k-1,k}^{(n+1)}-1}\frac{g_{k-1,i}^{(n+1)}}{g_{k-1,i}^{(n)}}$$

and

$$\lim_{n\to\infty}\frac{g_{k,i}^{(n+1)}}{g_{k,i}^{(n)}}=\frac{b_k-b_i}{b_k-b_i}\frac{b_k-1}{b_k-1}b_i=b_i$$

since $b_i\neq 1\ \forall i$ and $b_k\neq b_i\ \forall i>k$. ■

This theorem is a generalization of the condition (α) proved by Laurent [12] for the convergence of the diagonals and columns of Richardson extrapolation process. The conditions of Theorem 5 cannot be satisfied if $\lim_{n\to\infty} g_i(n)$ exists and is different from zero and infinity.

If the conditions of Theorem 5 are only verified for $i\leq k$ then we shall only deduce the convergence to S of the columns 0 to k from the theorem (but the conditions are only sufficient).

Let us now deal with convergence acceleration results.

Theorem 6. *Let us assume that the conditions of Theorem* 5 *are satisfied. If* $\lim_{n\to\infty}(E_{k-1}(S_{n+1})-S)/(E_{k-1}(S_n)-S)=b_k$ *then*

$$E_k(S_n)-S=o(E_{k-1}(S_n)-S).$$

Proof. We have:

$$\frac{E_k^{(n)}-S}{E_{k-1}^{(n)}-S}=\frac{g_{k-1,k}^{(n+1)}/g_{k-1,k}^{(n)}-(E_{k-1}^{(n+1)}-S)/(E_{k-1}^{(n)}-S)}{g_{k-1,k}^{(n+1)}/g_{k-1,k}^{(n)}-1}$$

and the result follows. ■

As it was the case for Theorem 4, this theorem is not very useful since it is difficult to check if the condition is satisfied or not. The next theorem will give conditions such that Theorem 6 will be true.

Theorem 7. *If the conditions of Theorem* 5 *are satisfied, if* $g_{i+1}(n)=o(g_i(n))\ \forall i$ *and if* $S_n=S+a_1g_1(n)+a_2g_2(n)+\ldots$ *then* $E_k(S_n)-S=o(E_{k-1}(S_n)-S)\ \forall k$.

Proof. If $S_n=S+a_1g_1(n)+a_2g_2(n)+\ldots$ then, by Theorem 3, we have

$$\frac{E_{k-1}^{(n+1)}-S}{E_{k-1}^{(n)}-S}=\frac{a_k g_{k-1,k}^{(n+1)}/g_{k-1,k}^{(n)}+a_{k+1}g_{k-1,k+1}^{(n+1)}/g_{k-1,k}^{(n)}+\cdots}{a_k+a_{k+1}g_{k-1,k+1}^{(n)}/g_{k-1,k}^{(n)}+\ldots}.$$

This ratio tends to b_k when n goes to infinity if $g_{k-1,i}^{(n)}=o(g_{k-1,k}^{(n)})$ for $i>k$ since:

$$g_{k-1,i}^{(n+1)}/g_{k-1,k}^{(n)}=g_{k-1,i}^{(n+1)}g_{k-1,i}^{(n)}/(g_{k-1,i}^{(n)}g_{k-1,k}^{(n)}).$$

Let us assume that $g_i(n)=o(g_k(n))\ \forall\, i>k$. We shall prove by induction that $g^{(n)}_{k-1,i}=o(g^{(n)}_{k-1,j})$ for $i>j\geqq k$. The property is true for $k=1$. We have:

$$\frac{g^{(n)}_{k,i}}{g^{(n)}_{k,j}}=\frac{g^{(n+1)}_{k-1,k}/g^{(n)}_{k-1,k}-g^{(n+1)}_{k-1,i}/g^{(n)}_{k-1,i}}{g^{(n+1)}_{k-1,k}/g^{(n)}_{k-1,k}-g^{(n+1)}_{k-1,j}/g^{(n)}_{k-1,j}}\,\frac{g^{(n)}_{k-1,i}}{g^{(n)}_{k-1,j}}.$$

Thus, by Theorem 5, we see that if the property is true for $k-1$ then it is still true for k and the result follows from Theorem 6. ■

This theorem needs some comments. A family of sequences satisfying $g_{i+1}(n)=o(g_i(n))$ forms what is called a scale of comparison and $a_1\,g_1(n)+a_2\,g_2(n)+\ldots$ is the asymptotic expansion of S_n-S with respect to this scale of comparison. Theorem 7 says that sequences whose error has an asymptotic expansion with respect to a given scale of comparison can be accelerated by an extrapolation process. It is obvious that some sequences have no asymptotic expansion with respect to a given scale of comparison but theorem 7 does not answer in that case. For example if $S_n=\lambda^n/n$ with $\lambda\in[-1,1[$ and if we choose $g_1(n)=\lambda^n$ then the conditions of Theorems 4 and 5 are satisfied and $E_1(S_n)$ converges:

$$E_1(S_n)=\frac{\lambda^{n+1}}{(\lambda-1)\,n(n+1)}.$$

The condition of Theorem 6 is verified and the convergence is accelerated since:

$$E_1(S_n)/S_{n+1}=1/n(\lambda-1).$$

The condition of Theorem 7 is not true since we cannot write

$$\lambda^n/n=a_1\,\lambda^n+\ldots$$

which shows that this condition is only sufficient. The same is true for the conditions given in Theorems 4 and 5. For example if $S_n=\lambda^n$ and $g_1(n)=n^{-1}$ then $E_1(S_n)=\lambda^n\,(\lambda(n+1)-n)$ which tends also to zero although the conditions of Theorems 4 and 5 do not hold. But, for this example, we cannot write $\lambda^n=a_1\,n^{-1}+\ldots$ and we have

$$\lim_{n\to\infty}|E_1(S_n)/S_{n+1}|=\infty.$$

Finally let us mention that if $b_k\neq 0\ \forall\, k$ in Theorems 6 and 7 then

$$E_k(S_n)-S=o(E_{k-1}(S_{n+1})-S).$$

The results obtained for the E-algorithm have to be compare with some results given by Germain-Bonne [8, p 33].

4. The Vector Case

Let us now consider the case where the sequence $\{S_n\}$ to be extrapolated is a sequence of vectors of $\mathbb{C}^p$ (or of a more general vector space). Such a case is very

important in numerical analysis since many iterative methods produce vector sequences. As in the first section, an extrapolation method will consist in assuming that (1) holds and computing $S \in \mathbb{C}^p$. The procedure to compute S has to be slightly modified since we are now dealing with vectors and since the sequences g_i are vector sequences. From (1) we have:

$$a_0 \Delta S_n + a_1 \Delta g_1(n) + \ldots + a_k \Delta g_k(n) = 0 \tag{4}$$

with $a_0 = -1$.

We shall first compute $a_0, a_1, \ldots, a_k$ and then obtain S by

$$S = -a_0 S_n - a_1 g_1(n) - \ldots - a_k g_k(n). \tag{5}$$

Let y be an arbitrary vector. Writting (4) for the indexes $n, \ldots, n+k-1$ and multiplying scalarly the equations by y we get:

$$\begin{pmatrix} 1 & 0 & \ldots & 0 \\ (y, \Delta S_n) & (y, \Delta g_1(n)) & \ldots & (y, \Delta g_k(n)) \\ \vdots & \vdots & & \vdots \\ (y, \Delta S_{n+k-1}) & (y, \Delta g_1(n+k-1)) & \ldots & (y, \Delta g_k(n+k-1)) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{6}$$

where Δ acts on the index n.

This system is assumed to be nonsingular. If the sequence $\{S_n\}$ has not the exact form (1) then the vector S obtained in this way will depend on n and k and we shall denote it by $E_k(S_n)$. From (6) and (5) it is easy to see that:

$$E_k(S_n) = \frac{\begin{vmatrix} S_n & g_1(n) & \ldots & g_k(n) \\ (y, \Delta S_n) & (y, \Delta g_1(n)) & \ldots & (y, \Delta g_k(n)) \\ \vdots & \vdots & & \vdots \\ (y, \Delta S_{n+k-1}) & (y, \Delta g_1(n+k-1)) & \ldots & (y, \Delta g_k(n+k-1)) \end{vmatrix}}{\begin{vmatrix} (y, \Delta g_1(n)) & \ldots & (y, \Delta g_k(n)) \\ \vdots & & \vdots \\ (y, \Delta g_1(n+k-1)) & \ldots & (y, \Delta g_k(n+k-1)) \end{vmatrix}}. \tag{7}$$

The numerator of (7) is a generalized determinant which denotes the vector obtained by expanding this determinant with respect to its first row by using the classical rules for expanding a determinant. The method for obtaining (7) is similar to the method used to define the vector generalization of Shanks transformation described in [2] which corresponds to the particular choice $g_i(n) = \Delta S_{n+i-1}$.

As in the scalar case there is a general recursive algorithm, the vector E-algorithm, to avoid the computation of the determinants in (7):

initializations $E_0^{(n)} = S_n \qquad n = 0, 1, \ldots$

$g_{0,i}^{(n)} = g_i(n) \quad i = 1, 2, \ldots$ and $n = 0, 1, \ldots$.

For $k = 1, 2, \ldots$ and $n = 0, 1, \ldots$

$$E_k^{(n)} = E_{k-1}^{(n)} - \frac{(y, \Delta E_{k-1}^{(n)})}{(y, \Delta g_{k-1,k}^{(n)})} g_{k-1,k}^{(n)},$$

$$g_{k,i}^{(n)} = g_{k-1,i}^{(n)} - \frac{(y, \Delta g_{k-1,i}^{(n)})}{(y, \Delta g_{k-1,k}^{(n)})} g_{k-1,k}^{(n)} \qquad i = k+1, k+2, \ldots.$$

In all these relations Δ operates on the index n; for example $\Delta g_{k,i}^{(n)} = g_{k,i}^{(n+1)} - g_{k,i}^{(n)}$.

Theorem 8.

$$E_k^{(n)} = E_k(S_n) \qquad k, n = 0, 1, \ldots.$$

Proof. We shall prove simultaneously that $g_{k,i}^{(n)}$ satisfies (7) where the first column of the numerator is replaced by $g_i(n)$, $(y, \Delta g_i(n)), \ldots, (y, \Delta g_i(n+k-1))$.

We set:

$$g_{k,i}^{(n)} = N_{k,i}^{(n)}/D_k^{(n)}; \qquad E_k^{(n)} = N_k^{(n)}/D_k^{(n)}.$$

From Sylvester determinantal identity we have

$$D_{k-1}^{(n)} N_k^{(n)} = N_{k-1}^{(n)} D_k^{(n)} - (-1)^{k-1} N_{k-1,k}^{(n)} A_k^{(n)}$$

where:

$$A_k^{(n)} = \begin{vmatrix} (y, \Delta S_n) & (y, \Delta g_1(n)) & \cdots & (y, \Delta g_{k-1}(n)) \\ (y, \Delta S_{n+k-1}) & (y, \Delta g_1(n+k-1)) & \ldots & (y, \Delta g_{k-1}(n+k-1)) \end{vmatrix}.$$

Thus

$$\frac{N_k^{(n)}}{D_k^{(n)}} = \frac{N_{k-1}^{(n)}}{D_{k-1}^{(n)}} - (-1)^{k-1} \frac{N_{k-1,k}^{(n)}}{D_{k-1}^{(n)}} \frac{A_k^{(n)}}{D_k^{(n)}}.$$

It remains to prove that $A_k^{(n)}/D_k^{(n)} = (-1)^{k-1}(y, \Delta E_{k-1}^{(n)})/(y, \Delta g_{k-1,k}^{(n)})$. From the relation of the vector E-algorithm and from the relation $E_k((y, S_n)) = (y, E_k(S_n))$ between the scalar and the vector extrapolation processes we have:

$$\frac{(y, \Delta E_{k-1}^{(n)})}{(y, \Delta g_{k-1,k}^{(n)})} = \frac{(y, E_{k-1}^{(n)} - E_k^{(n)})}{(y, g_{k-1,k}^{(n)})} = \frac{(y, N_{k-1}^{(n)}) D_k^{(n)} - (y, N_k^{(n)}) D_{k-1}^{(n)}}{D_k^{(n)} (y, N_{k-1,k}^{(n)})}.$$

From Sylvester identity we see that the numerator of the preceding relation is equal to $(-1)^{k-1} A_k^{(n)} (y, N_{k-1,k}^{(n)})$ which ends the proof. ■

By construction of the process we have:

Theorem 9. *If* $S_n = S + a_1 g_1(n) + \ldots + a_k g_k(n)$ $\forall n$ *then* $E_k(S_n) = S$ $\forall n$ *and* $\forall y$ *such that the denominators of* (7) *do not vanish.*

And more generally:

Theorem 10. *If* $S_n = S + a_1 g_1(n) + a_2 g_2(n) + \ldots$ *then* $E_k(S_n) = S + a_{k+1} g_{k,k+1}^{(n)} + a_{k+2} g_{k,k+2}^{(n)} + \ldots$ $\forall k, n$.

Proof. By induction. The property is true for $k = 0$. If it is true for $k-1$ then:

$$E_k^{(n)} = S + \sum_{i \geq k} a_i g_{k-1,i}^{(n)} - \sum_{i \geq k} a_i \frac{(y, \Delta g_{k-1,i}^{(n)})}{(y, \Delta g_{k-1,k}^{(n)})} g_{k-1,k}^{(n)}$$

$$= S + \sum_{i \geq k} a_i \left\{ g_{k-1,i}^{(n)} - \frac{(y, \Delta g_{k-1,i}^{(n)})}{(y, \Delta g_{k-1,k}^{(n)})} g_{k-1,k}^{(n)} \right\} = S + \sum_{i \geq k+1} a_i g_{k,i}^{(n)}$$

by the recurrence relation of $g_{k,i}^{(n)}$ and $g_{k,k}^{(n)} = 0$ $\forall n$. ■

5. Extensions

The work described above is susceptible of several extensions. First the study of the E-algorithm has to be continued: convergence and convergence acceleration

theorems, choice of the sequences g_i, numerical stability, singular cases, applications, ...

Then generalizations of the algorithm can be defined; for example an acceleration parameter can be introduced in the algorithm, its optimal value can be computed and then replaced by an approximation. The new algorithm thus obtained will be a generalization of the θ-algorithm [1] derived in the same way from the ε-algorithm and which is one of the most powerful acceleration method actually known [18]. This new algorithm will be called the T-algorithm and its rules will be:

$$T_0^{(n)} = S_n,$$

$$T_k^{(n)} = T_{k-1}^{(n+1)} - \frac{\Delta T_{k-1}^{(n+1)}}{\Delta D_k^{(n)}} D_k^{(n)}$$

with $D_k^{(n)} = -\dfrac{\Delta T_{k-1}^{(n)}}{\Delta g_{k-1,k}^{(n)}} g_{k-1,k}^{(n+1)}$ and the $g_{k,i}^{(n)}$ defined as in the E-algorithm.

Another possible generalization is the confluent case which is also defined in a way completely similar to the work of Wynn [21] for the ε-algorithm. The problem to deal with is to extrapolate a function f. f is assumed to satisfy:

$$f(t) = S + a_1 f_1(t) + \ldots + a_k f_k(t).$$

The extrapolated value of f will be S which is computed by solving the system:

$$\begin{aligned} f(t) &= S + a_1 f_1(t) + \ldots + a_k f_k(t) \\ f'(t) &= a_1 f_1'(t) + \ldots + a_k f_k'(t) \\ &\vdots \\ f^{(k)}(t) &= a_1 f_1^{(k)}(t) + \ldots + a_k f_k^{(k)}(t) \end{aligned}$$

In general the value of S thus obtained will depend on k and t and it will be denoted by $E_k(t)$. We have:

$$E_k(t) = \frac{\begin{vmatrix} f(t) & f'(t) & \ldots & f^{(k)}(t) \\ f_1(t) & f_1'(t) & \ldots & f_1^{(k)}(t) \\ \vdots & \vdots & & \vdots \\ f_k(t) & f_k'(t) & \ldots & f_k^{(k)}(t) \end{vmatrix}}{\begin{vmatrix} f_1'(t) & \ldots & f_1^{(k)}(t) \\ \vdots & & \vdots \\ f_k'(t) & \ldots & f_k^{(k)}(t) \end{vmatrix}}.$$

If $f_i(t) = f^{(i)}(t)$ then this is exactly the confluent form of the ε-algorithm [20]. If $f_i(t) = t^i$ then we obtain the Taylor expansion of f around 0 and for the choice $f_i(t) = g^{(i-1)}(t)$ where g is some given function, we get the G-transformation again [10].

The recursive computation of $E_k(t)$ can be done with the scalar E-algorithm applied to the sequence $\{S_n\}$ defined by $\Delta^n S_0 = f^{(n)}(t)$ with the sequences g_i given by $\Delta^n g_i(0) = f_i^{(n)}(t)$ and we shall obtain $E_k^{(0)} = E_k(t)$ [5].

Acknowledgment. I would like to thank M. Duc-Jacquet for pointing out to me the paper by Mühlbach.

References

1. Brezinski C (1971) Accélération de suites à convergence logarithmique. CR Acad Sc Paris, 273 A : 727-730
2. Brezinski C (1975) Généralisations de la transformation de Shanks, de la table de Padé et de l'ε-algorithme. Calcolo, 12:317-360
3. Brezinski C (1977) Accélération de la convergence en analyse numérique, (Lecture Notes in Mathematics vol 584). Springer, Berlin Heidelberg New York
4. Brezinski C (1978) Algorithmes d'accélération de la convergence. Etude numérique, Technip, Paris
5. Brezinski C (1979) Sur le calcul de certains rapports de déterminants. In: Wuytack L (ed) Padé approximation and its applications (Lecture Notes in Mathematics vol 765). Springer, Berlin Heidelberg New York
6. Cordellier F (1980) Analyse numérique des transformations de suites et de séries. Thèse, Université de Lille (in press)
7. Gantmacher FR (1960) The theory of matrices. Chelsea Publications, New York
8. Germain-Bonne B (1978) Estimation de la limite de suites et formalisation de procédés d'accélération de convergence. Thèse, Université de Lille
9. Gray HL, Atchison TA (1968) The generalized G-transform. Math Comput 22:595-606
10. Gray HL, Schucany WR (1969) Some limiting cases of the G-transformation. Math Comput 23:849-859
11. Håvie T (1979) Generalized Neville type extrapolation schemes. BIT 19:204-213
12. Laurent PJ (1964) Etude de procédés d'extrapolation en analyse numérique. Thèse, Université de Grenoble
13. Levin D (1973) Development of non-linear transformations for improving convergence of sequences. Internat J Comput Math B3:371-388
14. Mühlbach G (1978) The general Neville-Aitken algorithm and some applications. Numer Math 31:97-110
15. Pye WC, Atchison TA (1973) An algorithm for the computation of higher order G-transformation. SIAM J Numer Anal 10:1-7
16. Shanks D (1955) Nonlinear transformations of divergent and slowly convergent sequences. J Math Phys 34:1-42
17. Sidi A (1979) Some properties of a generalization of the Richardson extrapolation process. Technical report 142, Technion, Haifa, Israel
18. Smith DA, Ford WF (1979) Acceleration of linear and logarithmic convergence. SIAM J Numer Anal 16:223-240
19. Wynn P (1956) On a device for computing the $e_m(S_n)$ transformation. MTAC 10:91-96
20. Wynn P (1956) On a procrustean technique for the numerical transformation of slowly convergent sequences ans series. Proc Cambridge Phil Soc 52:663-671
21. Wynn P (1960) Confluent forms of certain nonlinear algorithms. Arch Math 11:223-234
22. Wynn P (1962) Acceleration technique in numerical analysis with particular reference to problems in one independant variable. Proc IFIP congress, North Holland, Amsterdam, p 149
23. Wynn P (1966) On the convergence and stability of the epsilon algorithm. SIAM J Numer Anal 3:91-122

Received September 9, 1979; Revised April 28, 1980