Particular rules for the Θ-algorithm

M. Redivo Zaglia

Dipartimento di Elettronica e Informatica, Università degli Studi di Padova, via Gradenigo 6 / a, I-35131 Padova, Italy

The Θ-algorithm is an extrapolation algorithm which can be very useful in accelerating some slowly convergent sequences. Like the other acceleration algorithms, the Θ-algorithm is quite sensitive to the propagation of rounding errors due to cancellation in the difference between two almost equal quantities.

In order to (partially) avoid this drawback, particular rules are given. They have to be used, instead of the usual rules of the algorithm, when two adjacent quantities in a column are nearly equal. Numerical examples show that these rules can improve the numerical stability of the algorithm in some cases while, in other cases, the improvement is non-existent.

AMS(MOS) subject classifications: 65B05, 65G05.

Keywords: Acceleration, extrapolation methods, rounding errors.

1. Introduction

Convergence acceleration methods are very useful tools which make it often possible to use sequences and series that converge slowly. These methods have been studied and developed for many years and they have a very wide range of applications in numerical analysis [4].

Many sequence transformations $T\colon (S_n) \to (T_n)$ are defined as a ratio of determinants.

For example, in the case of Shanks' transformation we have

$$T_n = e_k(S_n) = \frac{\begin{vmatrix} S_n & S_{n+1} & \cdots & S_{n+k} \\ \Delta S_n & \Delta S_{n+1} & \cdots & \Delta S_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta S_{n+k-1} & \Delta S_{n+k} & \cdots & \Delta S_{n+2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \Delta S_n & \Delta S_{n+1} & \cdots & \Delta S_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta S_{n+k-1} & \Delta S_{n+k} & \cdots & \Delta S_{n+2k-1} \end{vmatrix}}.$$

If the determinants involved in the definition of T_n have a special structure, then it is possible, by applying determinantal identities to the numerator and the denominator, to obtain some rules for computing them recursively without computing explicitly the determinants involved in their definition. These rules are called an extrapolation algorithm.

For implementing Shanks' transformation, it is possible to use the ϵ-algorithm of Wynn

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

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}}, \qquad k, n = 0, 1, \ldots,$$

and we have $\varepsilon_{2k}^{(n)} = e_k(S_n)$.

All the quantities computed by an extrapolation algorithm can be displayed in a double entry table as follows (in the case of the ε-algorithm)

$$\begin{array}{cccccc}
\varepsilon_{-1}^{(0)} = 0 & & & & & \\
 & \varepsilon_0^{(0)} = S_0 & & & & \\
\varepsilon_{-1}^{(1)} = 0 & & \varepsilon_1^{(0)} & & & \\
 & \varepsilon_0^{(1)} = S_1 & & \varepsilon_2^{(0)} & & \\
\varepsilon_{-1}^{(2)} = 0 & & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} & \\
\vdots & \varepsilon_0^{(2)} = S_2 & & \varepsilon_2^{(1)} & & \ddots \\
\vdots & \vdots & \varepsilon_1^{(2)} & & \varepsilon_3^{(1)} & \\
\vdots & \vdots & \vdots & \varepsilon_2^{(2)} & & \ddots \\
\vdots & \vdots & \vdots & \vdots & \varepsilon_3^{(2)} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Starting from the first two columns, $(\varepsilon_{-1}^{(n)} = 0)$ and $(\varepsilon_0^{(n)} = S_n)$, the rule of the ε-algorithm allows to proceed in this table from left to right and from top to bottom (normal form of an extrapolation algorithm).

The rule of the ε-algorithm relates quantities located at the four corners of a rhombus

$$\begin{tikzcd}
 & \varepsilon_k^{(n)} \arrow[dr] & \\
\varepsilon_{k-1}^{(n+1)} \arrow[ur] \arrow[dr] & & \varepsilon_{k+1}^{(n)} \\
 & \varepsilon_k^{(n+1)} \arrow[ur] &
\end{tikzcd}$$

and we saw that the numbers $\varepsilon_{2k}^{(n)}$ are approximations to the limit S_∞ of the sequence (S_n) while the $\varepsilon_{2k+1}^{(n)}$ are intermediate results. We are in fact not interested in the auxiliary quantities $\varepsilon_{2k+1}^{(n)}$ and thus we can use the so-called cross rule due to Wynn which only involves quantities with an even lower index.

Setting

$$\begin{array}{ccccc} & & N=\varepsilon_{2k}^{(n)} & & \\ & a=\varepsilon_{2k-1}^{(n+1)} & & b=\varepsilon_{2k+1}^{(n)} & \\ W=\varepsilon_{2k-2}^{(n+2)} & & C=\varepsilon_{2k}^{(n+1)} & & E=\varepsilon_{2k+2}^{(n)} \\ & e=\varepsilon_{2k-1}^{(n+2)} & & d=\varepsilon_{2k+1}^{(n+1)} & \\ & & S=\varepsilon_{2k}^{(n+2)} & & \end{array}$$

the cross rule is

$$(N-C)^{-1}+(S-C)^{-1}=(W-C)^{-1}+(E-C)^{-1},$$

that is, E can be computed by

$$E=C+\left[(N-C)^{-1}+(S-C)^{-1}-(W-C)^{-1}\right]^{-1}$$

with $\varepsilon_{-2}^{(n)}=\infty$.

2. Numerical instability

A crucial problem for every extrapolation method is the propagation of cancellation errors due to the finite accuracy of the computer. The better the extrapolation algorithm works, the more severe is the effect of rounding errors. For example, in the case of the ε-algorithm and the cross rule described in the preceding section, there are two main problems:

- If N is different from C but very close to it (both quantities are approximations of S), then a cancellation error arises in the computation of b (near-breakdown in the algorithm) which will be large and badly computed. If S is also different from C but very close to it, then d will be large and badly computed. Thus, in computing E, we have, in the denominator, the difference of two large and badly computed numbers. If $N=C$ then b is infinity (breakdown in the algorithm). If S is also equal to C then d is infinity. Thus E is undefined.
- If a is different from e but close to it, C will be large and badly computed. Thus b and d will be almost equal and E will be the algebraic sum of two large and badly computed numbers (near-breakdown in the algorithm). If $a=e$ then C is infinity (breakdown in the algorithm). If $N\neq C$ and $S\neq C$ then $b=d$ and E will be undefined.

There are two possible answers for avoiding numerical instability in an extrapolation algorithm: its progressive form and its particular rules.

3. Progressive forms

To obtain the progressive form for the ε-algorithm, we compute the first descending diagonal $(\varepsilon_k^{(0)})$ by the bordering method or by the block bordering method with the reverse bordering method [5]. Then we write the rule of the ε-algorithm as

$$\varepsilon_{k+1}^{(n+1)} = \varepsilon_{k+1}^{(n)} + \frac{1}{\varepsilon_{k+2}^{(n)} - \varepsilon_k^{(n+1)}}.$$

In general a progressive form allows, starting from the first diagonal $((\varepsilon_k^{(0)})$ in the ε-algorithm) and the second column $((\varepsilon_0^{(n)} = S_n)$ in the ε-algorithm) to compute all the other quantities in the table. Of course this rule still suffers from numerical instability since, when k is even, we have to compute the difference of two almost equal quantities. However, the instability is not so severe since, usually, $\varepsilon_{2k+2}^{(n)}$ is a better approximation of S_∞ than $\varepsilon_{2k}^{(n+1)}$ and both quantities have fewer digits in common than $\varepsilon_{2k}^{(n)}$ and $\varepsilon_{2k}^{(n+1)}$ (normal form of the ε-algorithm).

The progressive form of the cross rule of Wynn is

$$S = C + \left[(W-C)^{-1} + (E-C)^{-1} - (N-C)^{-1}\right]^{-1}.$$

4. Particular rules

The particular rules are obtained by modifying the rule of the algorithm in order to obtain a more stable algorithm.

For instance in Aitken's Δ^2 process we have

$$T_n = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n}, \quad n = 0, 1, \ldots.$$

This formula is highly numerically unstable since, if S_n, S_{n+1} and S_{n+2} are almost equal, cancellation errors arise both in the numerator and in the denominator and T_n is badly computed.

If we write

$$T_n = S_n - \frac{(S_{n+1} - S_n)^2}{S_{n+2} - 2S_{n+1} + S_n}, \quad n = 0, 1, \ldots,$$

we obtain a much more stable formula. Again, of course, cancellation errors arise in this formula when computing $(S_{n+1} - S_n)^2$ and $S_{n+2} - 2S_{n+1} + S_n$, but the term $(S_{n+1} - S_n)^2/(S_{n+2} - 2S_{n+1} + S_n)$ is a correction term to S_n which explains the better stability of this second formula.

Using the previous notations, the normal form of the ε-algorithm gives

$$C = W + 1/(e - a),$$
$$b = a + 1/(C - N),$$
$$d = e + 1/(S - C),$$
$$E = C + 1/(d - b).$$

If a is nearly equal to e, C will be large and badly computed (close to infinity), b and d will be almost equal and, as previously said, E will be undetermined.

After some algebraic manipulations, the cross rule of the ε-algorithm can be written as

$$E = r(1 + r/C)^{-1},$$

with

$$r = S(1 - S/C)^{-1} + N(1 - N/C)^{-1} - W(1 - W/C)^{-1}.$$

Wynn [11] proved that this rule can compute E in situations in which the normal rule given in section 1 fails. If C is infinity, which happens if $a = e$, this rule reduces to

$$E = S + N - W,$$

thus allowing the computation of E, if S, N, and W are defined, by jumping over the singularity. If the normal rules of the ε-algorithm would be used, $a = e$ would lead to C being infinity, which would then imply $b = d$. Hence, E could not be computed by the normal rule.

This rule is only valid when there is one isolated singularity, that is, when only two adjacent quantities in a column (a and e in our example) are equal or almost equal. Obviously it can also be applied for computing in a more stable way a quantity with a lower odd index when two adjacent quantities in an even column are equal or almost equal.

Wynn's particular rule was extended by Cordellier to the case of an arbitrary number of equal or nearly equal quantities in the ε-algorithm [6]. Wynn's particular rule can also be used for the first and second generalizations of the ε-algorithm and in the ρ-algorithm. Wynn also proposed a particular rule for the *qd*-algorithm. Using the notion of Schur complement, Brezinski [3] obtained particular rules for the E-algorithm and for some vector sequence transformations, such as the so-called RPA, CRPA and the H-algorithm, by computing directly the elements of the column $m + k$ of the table in terms of the elements of column k without computing the intermediate columns and then avoiding division by zero or numerical instability due to cancellation errors. Regarding the vector ε-algorithm, the normal cross rule is the same as that for the scalar

ε-algorithm and a particular rule was obtained by Cordellier [7] if N, S and W are different from C and if $(N-C)^{-1}+(S-C)^{-1}$ is not equal to $(W-C)^{-1}$.

5. The Θ-algorithm

Let me now recall the Θ-algorithm obtained by Brezinski [1],

$$\Theta_{-1}^{(n)}=0, \quad \Theta_0^{(n)}=S_n, \qquad n=0,1,\ldots,$$

$$\Theta_{2k+1}^{(n)}=\Theta_{2k-1}^{(n+1)}+\frac{1}{\Theta_{2k}^{(n+1)}-\Theta_{2k}^{(n)}}, \qquad k,n=0,1,\ldots,$$

$$\Theta_{2k+2}^{(n)}=\Theta_{2k}^{(n+1)}+\frac{\omega_k^{(n)}}{\Theta_{2k+1}^{(n+1)}-\Theta_{2k+1}^{(n)}}, \qquad k,n=0,1,\ldots,$$

with

$$D_k^{(n)}=\left(\Theta_k^{(n+1)}-\Theta_k^{(n)}\right)^{-1},$$

$$\omega_k^{(n)}=-\frac{\Delta\Theta_{2k}^{(n+1)}}{\Delta D_{2k+1}^{(n)}}=-\frac{\Theta_{2k}^{(n+2)}-\Theta_{2k}^{(n+1)}}{\left(\Theta_{2k+1}^{(n+2)}-\Theta_{2k+1}^{(n+1)}\right)^{-1}-\left(\Theta_{2k+1}^{(n+1)}-\Theta_{2k+1}^{(n)}\right)^{-1}}.$$

As for the ε-algorithm, the quantities belonging to the odd columns are intermediate results.

The numerical experiments conducted by Smith and Ford [8,9] show that the Θ-algorithm is among the algorithms which provide a good answer in the greatest number of cases.

The Θ-algorithm is also a quite powerful method for summing some divergent series, as exemplified in [2,10].

6. Particular rules for Θ-algorithm

The rule of the Θ-algorithm for quantities belonging to odd columns, relates, as in the ε-algorithm, quantities located at the four corners of a rhombus,

$$\begin{tikzcd}
 & \Theta_{2k}^{(n)} \arrow[dr] & \\
\Theta_{2k-1}^{(n+1)} \arrow[ur] \arrow[dr] & & \Theta_{2k+1}^{(n)} \\
 & \Theta_{2k}^{(n+1)} \arrow[ur] &
\end{tikzcd}$$

but, in the case of even columns, there are much more quantities involved in the rule

$$\begin{tikzcd}
 & \Theta_{2k+1}^{(n)} \arrow[dr] & \\
\Theta_{2k}^{(n+1)} \arrow[ur] \arrow[dr] & & \Theta_{2k+2}^{(n)} \\
 & \Theta_{2k+1}^{(n+1)} \arrow[ur] & \\
\Theta_{2k}^{(n+2)} \arrow[ur] \arrow[dr] & & \\
 & \Theta_{2k+1}^{(n+2)} \arrow[uuur] &
\end{tikzcd}$$

Thus, we shall have to consider separately two cases according to the presence of two adjacent equal or nearly equal quantities in the same column, which may be odd or even.

6.1. RULE FOR EVEN COLUMNS

We set

$$\begin{array}{ccccc}
 & & N=\Theta_{2k+1}^{(n)} & & \\
 & a=\Theta_{2k}^{(n+1)} & & b=\Theta_{2k+2}^{(n)} & \\
W=\Theta_{2k-1}^{(n+2)} & & C=\Theta_{2k+1}^{(n+1)} & & E=\Theta_{2k+3}^{(n)} \\
 & e=\Theta_{2k}^{(n+2)} & & d=\Theta_{2k+2}^{(n+1)} & \\
T=\Theta_{2k-1}^{(n+3)} & & S=\Theta_{2k+1}^{(n+2)} & & \\
 & i=\Theta_{2k}^{(n+3)} & & & \\
 & & V=\Theta_{2k+1}^{(n+3)} & &
\end{array}$$

Using the normal form of the algorithm we have

$$C=W+\frac{1}{e-a},$$

$$b=a+\frac{\omega_k^{(n)}}{C-N}, \quad \text{with } \omega_k^{(n)}=-\frac{e-a}{(S-C)^{-1}-(C-N)^{-1}},$$

$$d=e+\frac{\omega_k^{(n+1)}}{S-C}, \quad \text{with } \omega_k^{(n+1)}=-\frac{i-e}{(V-S)^{-1}-(S-C)^{-1}},$$

$$E=C+\frac{1}{d-b}.$$

If a and e belong to an even column and E belongs to an odd column, and if a is different from e but close to it, C will be large and badly computed. If N and S are not close to C, that is, if they are not large (isolated singularity), then $\omega_k^{(n)}$ will be the ratio of two small and badly computed quantities and thus b will be badly computed. If also V is not large and i is not close to e, then d is nearly equal to e. Thus E will be computed from badly computed quantities. Moreover, if i is close to a and e then the situation becomes worse because S is also large and badly computed and almost all the quantities involved in the computation of E are badly computed.

We shall now make some algebraic manipulations, similar to those of Wynn, with the intention of obtaining a formula for E that will possibly be more stable.

$$\begin{aligned}
E &= C + \frac{1}{d-b} \\
&= C + \frac{1}{e + \omega_k^{(n+1)}(S-C)^{-1} - a - \omega_k^{(n)}(C-N)^{-1}} \\
&= C + \frac{1}{(C-W)^{-1} + \omega_k^{(n+1)}(S-C)^{-1} - \omega_k^{(n)}(C-N)^{-1}} \\
&= C + \frac{C}{C(C-W)^{-1} - C\omega_k^{(n+1)}(C-S)^{-1} - C\omega_k^{(n)}(C-N)^{-1}} \\
&= C + \frac{C}{(C-W+W)(C-W)^{-1} - \omega_k^{(n+1)}(C-S+S)(C-S)^{-1} - \omega_k^{(n)}(C-N+N)(C-N)^{-1}} \\
&= C + \frac{C}{1 + W(C-W)^{-1} - \omega_k^{(n+1)}[1 + S(C-S)^{-1}] - \omega_k^{(n)}[1 + N(C-N)^{-1}]} \\
&= C - \frac{C}{\omega_k^{(n+1)} + \omega_k^{(n)} - 1 - W(C-W)^{-1} + \omega_k^{(n+1)}S(C-S)^{-1} + \omega_k^{(n)}N(C-N)^{-1}} \\
&= \frac{C(\omega_k^{(n+1)} + \omega_k^{(n)} - 1) - C - CW(C-W)^{-1} + \omega_k^{(n+1)}CS(C-S)^{-1} + \omega_k^{(n)}CN(C-N)^{-1}}{\omega_k^{(n+1)} + \omega_k^{(n)} - 1 - W(C-W)^{-1} + \omega_k^{(n+1)}S(C-S)^{-1} + \omega_k^{(n)}N(C-N)^{-1}} \\
&= \frac{C(\omega_k^{(n+1)} + \omega_k^{(n)} - 2) + \omega_k^{(n+1)}S(1-S/C)^{-1} + \omega_k^{(n)}N(1-N/C)^{-1} - W(1-W/C)^{-1}}{\omega_k^{(n+1)} + \omega_k^{(n)} - 1 + C^{-1}[\omega_k^{(n+1)}S(1-S/C)^{-1} + \omega_k^{(n)}N(1-N/C)^{-1} - W(1-W/C)^{-1}]} \\
&= \frac{C(\omega_k^{(n+1)} + \omega_k^{(n)} - 2) + \omega_k^{(n+1)}S(1-S/C)^{-1} + \omega_k^{(n)}N(1-N/C)^{-1} - W(1-W/C)^{-1}}{1 + C^{-1}[C(\omega_k^{(n+1)} + \omega_k^{(n)} - 2) + \omega_k^{(n+1)}S(1-S/C)^{-1} + \omega_k^{(n)}N(1-N/C)^{-1} - W(1-W/C)^{-1}]}.
\end{aligned}$$

If we set

$$\begin{aligned}
r = {} & C(\omega_k^{(n+1)} + \omega_k^{(n)} - 2) + \omega_k^{(n+1)}S(1-S/C)^{-1} + \omega_k^{(n)}N(1-N/C)^{-1} \\
& - W(1-W/C)^{-1},
\end{aligned}$$

we obtain an expression of the same form as the particular rule of Wynn

$$E = r(1 + r/C)^{-1}.$$

Now we shall modify also the computation of the quantities $\omega_k^{(n)}$ and $\omega_k^{(n+1)}$

$$\omega_k^{(n)} = -\frac{e-a}{(S-C)^{-1}-(C-N)^{-1}} = \frac{(C-W)^{-1}}{(C-S)^{-1}+(C-N)^{-1}}$$
$$= \frac{(1-W/C)^{-1}}{(1-S/C)^{-1}+(1-N/C)^{-1}},$$
$$\omega_k^{(n+1)} = -\frac{i-e}{(V-S)^{-1}-(S-C)^{-1}} = \frac{(S-T)^{-1}}{(S-V)^{-1}+(S-C)^{-1}}$$
$$= \frac{(1-T/S)^{-1}}{(1-V/S)^{-1}+(1-C/S)^{-1}}.$$

Using the preceding formulae for the ω's we have

$$r = C\left(\omega_k^{(n+1)}+\omega_k^{(n)}-2\right) - \frac{C(1-V/S)}{(1-T/S)(2-C/S-V/S)}$$
$$+\frac{N(1-S/C)}{(1-W/C)(2-N/C-S/C)} - \frac{W}{(1-W/C)}$$
$$= C\left[\omega_k^{(n+1)}+\omega_k^{(n)}-2-\frac{(1-V/S)}{(1-T/S)(2-C/S-V/S)}\right]$$
$$+\frac{N(1-S/C)}{(1-W/C)(2-N/C-S/C)} - \frac{W}{(1-W/C)}$$

and

$$r/C = \omega_k^{(n+1)}+\omega_k^{(n)}-2-\frac{(1-V/S)}{(1-T/S)(2-C/S-V/S)}$$
$$+\frac{N/C(1-S/C)}{(1-W/C)(2-N/C-S/C)} - \frac{W/C}{(1-W/C)}.$$

In the case of one isolated singularity (for example $a=e$), we have $C=\infty$. If $T \neq S$, $V \neq S$ and $S \neq 0$, it follows from the particular rule

$$\omega_k^{(n)} = 1/2 \quad \text{and} \quad \omega_k^{(n+1)} = \frac{(1-T/S)^{-1}}{(1-V/S)^{-1}} \neq \infty,$$
$$r = \alpha C + \beta$$

where

$$\alpha = \omega_k^{(n)} + \omega_k^{(n+1)} - 2 \neq \infty \quad \text{and} \quad \beta = N/2 - W \neq \infty,$$
$$r/C = \alpha$$

and then

$$E = \infty.$$

In practice, it is very difficult to have only one isolated singularity due to convergence because often when we find, in an even column, two almost equal quantities, then on the same column all the other successive quantities are also nearly equal to the first two. In this case, the preceding rule does not permit to avoid completely the unstable computations. For instance, when $a = e = i$ then $C = S = \infty$ and when computing the ω's and r we have the undetermined ratio S/C.

However, with the exception of the computation of $\Theta_1^{(n)}$, the use of the special rule permits to jump over the singularity. In fact, when the program implementing the Θ-algorithm finds two adjacent quantities in an even column exactly equal (due to the finite precision of the computer), it has to stop because a division by zero in the next odd column occurs. With this particular rule a quantity in an odd column can always be computed because the formulae relate only elements belonging to the odd columns and these elements are usually very large (when the preceding even column contains good approximations of the limit) but almost never equal.

6.2. RULE FOR ODD COLUMNS

We set

$$\begin{array}{ccccc}
 & & N = \Theta_{2k}^{(n)} & & \\
 & a = \Theta_{2k-1}^{(n+1)} & & b = \Theta_{2k+1}^{(n)} & \\
W = \Theta_{2k-2}^{(n+2)} & & C = \Theta_{2k}^{(n+1)} & & E = \Theta_{2k+2}^{(n)} \\
 & e = \Theta_{2k-1}^{(n+2)} & & d = \Theta_{2k+1}^{(n+1)} & \\
T = \Theta_{2k-2}^{(n+3)} & & S = \Theta_{2k}^{(n+2)} & & \\
 & i = \Theta_{2k-1}^{(n+3)} & & h = \Theta_{2k+1}^{(n+2)} & \\
 & & V = \Theta_{2k}^{(n+3)} & &
\end{array}$$

Using the normal form of the algorithm we have

$$C = W + \frac{\omega_{k-1}^{(n+1)}}{e-a} \quad \text{with} \quad \omega_{k-1}^{(n+1)} = -\frac{T-W}{(i-e)^{-1} - (e-a)^{-1}},$$

that is,

$$C = W - \frac{T-W}{(e-a)(i-e)^{-1} - 1},$$

$$b = a + \frac{1}{C-N}, \quad d = e + \frac{1}{S-C}, \quad h = i + \frac{1}{S-V},$$

$$E = C + \frac{\omega_k^{(n)}}{d-b} \quad \text{with} \quad \omega_k^{(n)} = -\frac{S-C}{(h-d)^{-1} - (d-b)^{-1}},$$

that is,

$$E = C - \frac{S-C}{(d-b)(h-d)^{-1} - 1}.$$

If a is different from e but close to it, then the denominator of C will be nearly equal to -1, and C will be almost equal to T. If e is different from i but close to it, then the denominator of C will be large. In addition, C will be almost equal to W. In this case, instability is not a real problem for the computation of C and all the other quantities. The only unstable condition occurs when a, e and i are all nearly equal, but this possibility almost never happens in practice for adjacent quantities in an odd column.

However, the normal form of the Θ-algorithm for the computation of quantities in even columns cannot be used if e and i (or h and d) are exactly equal. Thus we look again for a different formula that makes it possible to avoid such a breakdown.

We make some algebraic manipulations and we obtain

$$\begin{aligned} C &= W - \frac{T-W}{(e-a)(i-e)^{-1} - 1} \\ &= W - \frac{(T-W)(i-e)}{(e-a)-(i-e)} \\ &= \frac{W(2e-a-i) - (T-W)(i-e)}{(2e-a-i)} \\ &= \frac{W(e-a) - T(i-e)}{(2e-a-i)}. \end{aligned}$$

If the two differences $e-a$ and $i-a$ are not almost equal, this formula also partially avoids the cancellation error of the normal form, when the even quantities W and T become very close.

7. Numerical results

We define a value m representing the number of decimal digits that the user allows to lose due to the cancellation error.

The particular rule for even columns of the Θ-algorithm shall be used for computing $\Theta_{2k+3}^{(n)}$ when, for any $k = 0, 1, \ldots$ and for any $n = 0, 1, \ldots,$ the following relations hold

$$\left|\frac{\Theta_{2k}^{(n+2)} - \Theta_{2k}^{(n+1)}}{\Theta_{2k}^{(n+1)}}\right| < 10^{-m} \quad \text{or} \quad \left|\frac{\Theta_{2k+2}^{(n+1)} - \Theta_{2k+2}^{(n)}}{\Theta_{2k+2}^{(n)}}\right| < 10^{-m}.$$

In our case we apply the rule also when there are several singularities, that is, when more than two adjacent quantities are nearly equal according to the preceding inequalities.

The particular rule for odd columns shall be used for computing $\Theta_{2k+2}^{(n)}$ when

$$\left|\frac{\Theta_{2k+1}^{(n+2)} - \Theta_{2k+1}^{(n+1)}}{\Theta_{2k+1}^{(n+1)}}\right| < 10^{-m} \quad \text{or} \quad \left|\frac{\Theta_{2k}^{(n+2)} - \Theta_{2k}^{(n+1)}}{\Theta_{2k}^{(n+1)}}\right| < 10^{-m}.$$

In all the following examples the number of exact digits ($-\log_{10}|$relative error$|$) is shown. The value 999.0 denotes that all the digits are exact. When a value is missing, it means that a breakdown occurs in the algorithm and the corresponding entry of the table cannot be computed. The sequence of values given for $n = 0, 1, 2, \ldots$ is $\Theta_0^{(0)}, \Theta_0^{(1)}, \Theta_0^{(2)}, \Theta_2^{(0)}, \Theta_2^{(1)}, \Theta_2^{(1)}, \Theta_2^{(2)}, \Theta_4^{(0)}, \Theta_4^{(1)}, \ldots.$

The computations have been made on a PC with the Microsoft FORTRAN Optimizing Compiler and we have a precision of about 15–16 decimal digits.

7.1. EXAMPLE 1

We consider the series

$$e^x = \sum_{i=0}^{\infty} x^i/i!$$

and its sequence of partial sums

$$S_0 = 0,$$

$$S_n = \sum_{i=0}^{n-1} x^i/i!, \quad n = 1, 2, \ldots.$$

This example was already considered by Wynn [11] and its interest is due to the fact that it presents, when $x = 2$, one isolated singularity in an odd column ($\Theta_1^{(1)} = \Theta_1^{(2)} = 0.5$) and another one when $x = 4$ ($\Theta_1^{(3)} = \Theta_1^{(4)} = 9.3749999999999 \times 10^{-2}$).

With $m = 12$ we obtain the results given in table 1.

For all values of x, when two successive values of the given sequence are not exactly equal, the particular rules allow the computation of the whole Θ-array. The normal rules instead run into a breakdown in the computation of some values. Regarding the stability, when both values (without and with particular rules) are computable, the particular rules give almost always the same number

Table 1

n	$x=-4.5$		$x=-2.0$		$x=2.0$		$x=2.5$		$x=4.0$	
	Θ	Θ p.r.	Θ	Θ p.r.	Θ	Θ p.r.	Θ	Θ p.r.	Θ	Θ p.r.
3	−0.68	−0.68	0.32	0.32		0.06	0.02	0.02	0.00	0.00
4	−0.85	−0.85	0.75	0.75	0.49	0.49	0.22	0.22	0.02	0.02
5	−0.84	−0.84	1.26	1.26	1.28	1.28	0.80	0.80		0.12
6	0.79	0.79	3.02	3.02		1.59	0.60	0.60		0.00
7	1.17	1.17	3.92	3.92	3.67	3.67	2.53	2.53		0.17
8	1.58	1.58	4.78	4.78	4.47	4.47	3.73	3.73		1.16
9	4.11	4.11	6.98	6.98		4.82	4.44	4.44		0.61
10	4.76	4.76	8.20	8.20	6.55	6.55	5.17	5.17		3.63
11	5.36	5.36	9.49	9.49	9.16	9.16	7.10	7.10		4.02
12	7.27	7.27	11.56	11.56		10.25	8.20	8.20		3.64
13	8.14	8.14	12.33	12.33	11.10	11.10	10.21	10.21		4.46
14	9.14	9.14	12.76	12.76	11.67	11.67	10.82	10.82		8.29
15	11.68	11.68	13.21	13.21		12.10	9.46	9.46		9.70
16	11.61	11.61	12.93	12.92	12.12	12.12	11.53	11.53		11.68
17	11.77	11.77	15.21	15.09	12.53	12.53	11.98	11.98		10.06
18	12.29	12.29	15.39	15.39		11.80	12.02	12.02		10.07
19	11.59	11.59		999.0		13.22	11.99	11.99		10.07
20	13.32	13.32		999.0		14.31	12.07	12.07		10.07
21	12.66	12.66		999.0		14.97	12.01	12.01		10.07
22		15.81		999.0		999.0		11.80		10.06
23		15.81		999.0		999.0		13.45		10.22
24		15.81		999.0		999.0		12.16		10.07

of exact digits. There are many cases where the particular rules give a better result (for example $\Theta_4^{(15)}$, $\Theta_4^{(18)}$, $\Theta_6^{(11)}$, $\Theta_6^{(12)}$, $\Theta_6^{(13)}$, $\Theta_8^{(8)}$, $\Theta_8^{(9)}$ and $\Theta_{10}^{(6)}$, for $x = 2.5$) and some cases where they are worse (for example $\Theta_6^{(11)}$, $\Theta_{10}^{(1)}$ and $\Theta_{10}^{(2)}$ for $x = -2.0$).

7.2. EXAMPLE 2

We consider the following sequence converging to $S_\infty = 1$:

$$S_0 = a,$$

$$S_{n+1} = \sqrt{S_n}, \quad n = 0, 1, \ldots.$$

We obtain for different values of a and m the results of table 2.

Again the particular rules make it possible to jump over all the breakdowns found with the normal rules. Moreover, they often provide better approximations to the limit.

When $a = 5$, the particular rules cannot continue beyond $n = 27$ because there is a breakdown in the particular rules for even columns due to the fact that r/C is exactly equal to -1. Moreover, with the same value of a, for $n = 16$

Table 2

n	$a=5$			$a=50$			$a=500$			$a=5000$		
	Θ	Θ p.r. $m=5$	Θ p.r. $m=12$	Θ	Θ p.r. $m=5$	Θ p.r. $m=12$	Θ	Θ p.r. $m=5$	Θ p.r. $m=12$	Θ	Θ p.r. $m=5$	Θ p.r. $m=12$
12	11.05	11.05	11.05	7.83	7.83	7.83	6.86	6.86	6.86	6.57	6.57	6.57
13	12.45	12.45	12.45	10.33	10.33	10.33	8.68	8.68	8.68	7.90	7.90	7.90
14	13.73	13.73	13.73	12.22	12.22	12.22	11.20	11.20	11.20	10.05	10.05	10.05
15	14.84	14.91	14.91	13.30	13.31	13.30	12.32	12.32	12.32	11.54	11.54	11.54
16	10.63	12.23	12.70	14.88	14.75	14.88	13.37	13.37	13.37	13.14	13.13	13.14
17	14.61	14.61	14.61	13.36	13.28	13.33	14.07	14.06	14.09	13.47	13.45	13.47
18	11.64	13.26	13.68	13.43	13.27	13.38	14.51	14.48	14.51	13.92	13.90	13.91
19	14.81	14.81	14.81		14.40	14.17	15.11	15.11	14.91	14.44	14.45	14.44
20	14.51	14.45	14.57		14.44	14.45	14.45	14.81	15.26	14.51	14.51	14.51
21	14.48	14.48	14.51		14.41	14.45	14.65	15.05	15.65	14.48	14.48	14.48
22	14.33	14.22	14.42		14.46	14.46	15.11	15.65	15.35	14.56	14.57	14.56
23		15.35	15.18		14.45	14.46	14.95	15.18	15.35	15.11	15.26	15.18
24		15.35	15.18		14.44	14.46	15.00	15.26	999.0		14.91	14.78
25		15.65	15.65		14.45	14.46	14.33	15.35	13.66		15.35	15.35
26		15.65	999.0		14.45	14.45	13.31	13.78	12.63		15.35	15.05
27		15.65	999.0		14.45	14.46	14.07	14.31	13.12		999.0	15.95

and $n = 18$, respectively, we have a good improvement of the number of decimal digits that are exact.

7.3. EXAMPLE 3

We consider the series

$$\frac{\pi^2}{6} = \sum_{i=0}^{\infty} (i+1)^{-2}$$

and the following sequence derived from the series

$$S_n = \sum_{i=0}^{n} (i+1)^{-2}, \quad n = 0, 1, \ldots .$$

For different choices of m we obtain the results of table 3.

As we can see, this example is very sensitive to the choice of m. If $m = 5$ or less, then both particular rules are applied nearly from the beginning for computing the values after the tenth of the third column and they continue to be used in the successive computations for all the values of the array. This behaviour permits to the algorithm to obtain in the first three values of the columns 18, 19, 20 and 21 (corresponding to $n = 27$ to $n = 38$) a number of digits more stable than that of the normal rules. Moreover, the results obtained with the particular rules (also if in some cases they are a little bit worse) do not

Table 3

n	Θ	Θ part. rules				n	Θ	Θ part. rules			
		$m=5$	$m=6$	$m=7$	$m=9$			$m=5$	$m=6$	$m=7$	$m=9$
27	8.57	8.49	8.57	8.57	8.57	63	−3.81	4.38	5.54	−3.82	−3.83
28	8.05	8.29	8.05	8.05	8.05	64	−5.19	7.88	8.16	−5.18	−5.19
29	7.23	8.59	7.23	7.23	7.23	65	−6.12	6.84	7.45	−6.09	−6.11
30	7.50	8.37	7.50	7.50	7.50	66	−6.33	7.24	7.53	−6.31	−6.33
31	6.77	8.73	6.77	6.77	6.77	67	−7.27	6.23	7.00	−7.22	−7.25
32	7.65	8.73	7.65	7.65	7.65	68	−8.55	5.20	5.61	−8.49	−8.52
33	6.34	8.73	6.34	6.34	6.34	69	−8.41	5.29	5.95	−8.35	−8.38
34	7.70	8.73	7.55	7.70	7.70	70	−9.70	4.68	5.07	−9.62	−9.65
35	7.97	8.73	7.27	7.98	7.98	81	4.22	3.96	3.97	5.90	6.02
36	7.78	8.73	7.42	7.79	7.79	72	−10.85	4.28	4.66	−10.74	−10.78
37	8.14	8.73	7.12	8.15	8.15	73	3.89	3.65	3.66	5.69	5.89
38	9.98	8.73	8.07	9.94	9.94	74	3.48	3.45	3.45	5.63	5.73
39	8.29	8.73	6.95	8.31	8.31	75	3.57	3.37	3.37	5.65	5.77
40	8.73	8.72	6.68	8.74	8.74	76	3.16	3.17	3.17	4.55	5.61
41	7.85	7.21	4.87	7.86	7.86	77	2.62	2.68	2.68	2.78	5.39
42	8.06	8.71	5.36	8.07	8.07	78	2.83	2.91	2.90	3.12	5.49
43	7.39	6.91	3.56	7.40	7.40	79	2.29	2.42	2.42	1.43	5.28
44	6.77	6.92	2.33	6.77	6.77	80	1.58	1.87	1.87	−0.45	5.01
45	7.10	6.93	2.23	7.10	7.10	81	1.97	2.16	2.16	0.08	5.18
46	6.63	6.75	0.90	6.63	6.63	82	1.26	1.61	1.61	−1.80	4.97
47	6.48	6.91	−0.85	6.48	6.48	83	4.76	4.53	4.53	−3.32	4.71
48	6.67	6.82	−0.50	6.61	6.61	84	0.93	1.35	1.35	−3.15	5.09
49	6.48	6.51	−2.24	6.48	6.48	85	4.30	4.16	4.16	−4.67	4.24
50	6.42	6.80	6.92	6.42	6.42	86	3.75	3.19	3.19	−5.79	3.73
51	6.42	6.96	−3.60	6.41	6.41	87	3.83	3.75	3.75	−6.02	3.77
52	6.13	6.91	7.04	6.07	6.07	88	3.29	2.75	2.75	−7.15	3.26
53	4.47	7.11	7.26	4.45	4.45	89	2.59	0.93	0.93	−8.77	2.58
54	4.72	7.03	7.17	4.68	4.68	90	2.82	2.32	2.32	−8.50	2.80
55	3.28	7.25	7.40	3.25	3.25	91	2.12	0.50	0.50	−10.12	2.12
56	1.56	8.85	8.02	1.52	1.52	92	0.55	4.15	4.15	−12.57	0.55
57	2.10	7.40	7.51	2.07	2.07	93	1.66	0.06	0.06	−11.47	1.65
58	0.41	7.89	7.33	0.37	0.37	94	0.09	3.90	3.90	−13.93	0.08
59	−1.52	6.67	6.69	−1.55	−1.55	95	−0.28	3.65	3.65	−15.21	−0.29
60	−0.74	6.83	6.74	−0.78	−0.78	96	−0.38	3.65	3.65	−15.28	−0.38
61	−2.66	5.54	6.11	−2.69	−2.69	97	−0.75	3.40	3.40	−16.56	−0.75
62	−4.04	7.99	8.35	−4.05	−4.06	98	−1.78	3.10	3.10	−18.59	−1.79

suffer from the instability present in some areas of the array and they give an acceptable approximation also for greater values of n (for example when $n = 96$ which corresponds to $\Theta_{64}^{(0)}$).

If $m = 6$ the particular rules are applied later, starting after the twentieth element of the third column. Thus in columns 18, 19, 20 and 21 we do not find any improvement and on the contrary the particular rules produce an area of instability near $n = 48$ which is not present with the normal rules. But in this case the particular rules are able to recover the results as in the case of $m = 5$.

If $m = 7$ or $m = 9$, due to the late application of the particular rules (for $m = 9$, for example, they are used only from column 5) the algorithm with particular rules behaves almost chaotically with some values better than the normal ones and other values very worse.

For any choices of m the number of exact digits of the results obtained in columns 2 to 16, with and without particular rules, are almost the same.

8. Conclusions

The existing theory of the Θ-algorithm does not permit to construct examples which must contain, in a column of the array, algebraic known values or an isolated singularity in a column of the array. This is possible for other extrapolation algorithms, for example for the ε-algorithm. The only possibility is to compare the results with those obtained using greater precision or symbolic computation. Doing that for some sequences, we have remarked that the values of column 1 (for which particular rules cannot be used) were yet affected by an error and thus in some cases the particular rules were not able to correct the instability of the results.

The main interest of using the particular rules is to avoid the breakdown of the normal rules (except when the breakdown occurs in column 1, when two values of the sequence of input data are exactly equal, and the unlucky case $r/C = -1$).

Regarding the gain in decimal digits, in general it is not really impressive and it is probably due also to the fact that the singularities are not isolated but there are areas (sometimes wide) of adjacent singularities and in this case the particular rules are not applicable (in theory).

References

[1] C. Brezinski, Accélération de suites à convergence logarithmique, C.R. Acad. Sci. Paris 273 (1971) A: 727–730.

[2] C. Brezinski, *Algorithmes d'Accélération de la Convergence – Etude Numérique* (Editions Technip, Paris, 1978).

[3] C. Brezinski, Other manifestations of the Schur complement, Lin. Alg. Appl. 111 (1988) 231–247.

[4] C. Brezinski and M. Redivo Zaglia, *Extrapolation Methods. Theory and Practice* (North-Holland, Amsterdam, 1991).

[5] C. Brezinski, M. Morandi Cecchi and M. Redivo Zaglia, The reverse bordering method, SIAM J. Matrix Anal. Appl., to appear.

[6] F. Cordellier, Démonstration algébrique de l'extension de l'identité de Wynn aux tables de Padé non normales, in: *Padé Approximation and its Applications*, ed. L. Wuytack, Lecture Notes in Mathematics vol. 765 (Springer, Berlin, 1979) pp. 36–60.
[7] F. Cordellier, Particular rules for the vector ε-algorithm, Numer. Math. 27 (1977) 203–207.
[8] D.A. Smith and W.F. Ford, Acceleration of linear and logarithmic convergence, SIAM J. Numer. Anal. 16 (1979) 223–240.
[9] D.A. Smith and W.F. Ford, Numerical comparison of nonlinear convergence accelerators, Math. Comput. 38 (1982) 481–499.
[10] E.J. Weniger and J. Čížek, Rational approximations for the modified Bessel function of the second kind, Comput. Phys. Commun. 59 (1990) 471–493.
[11] P. Wynn, Singular rules for certain nonlinear algorithms, BIT 3 (1963) 175–195.