Convergence acceleration for boundary value problems with singularities using the E-algorithm

Pedro M. Lima*, Mário M. Graça

Departamento de Matemática, Instituto Superior Técnico, Av. Rovisco Pais, 1096 Lisboa Codex, Portugal

Received 21 June 1993; revised 21 March 1994

Abstract

In the present work we use the E-algorithm to accelerate the convergence of finite-difference schemes for ordinary differential equations. As a model, a boundary-value problem for a second-order differential equation is considered, where the right-hand side may have different kinds of singularities. An asymptotic error expansion is obtained, enabling the use of the E-algorithm to accelerate the convergence. The applicability and the efficiency of the E-algorithm are discussed and illustrated by numerical examples.

Keywords: Convergence acceleration; Boundary-value problem; Finite-difference method; Green function; Richardson extrapolation; E-algorithm; Trapezoidal rule

1. Introduction

Extrapolation methods are a powerful way for obtaining numerical results of high accuracy with little computational effort. Particularly in the case of differential equations it is well known that Richardson extrapolation [24], under certain conditions, improves considerably the accuracy of the results obtained by finite-difference methods. The applicability of Richardson extrapolation was widely discussed, in particular in [5, Chs. 2 and 3; 20, Ch. 3]. Basically, the error of the finite-difference method must be representable under the form:

$$u^h(x) - u(x) = \sum_{k=1}^{m-1} C_k(x) h^k + r^h(x)$$  (1.1)

where $u^h(x)$ is the approximate solution, $u(x)$ is the exact solution, $C_k(x)$ are functions independent of $h$; $r^h(x)$ is a function such that $\|r^h\| = O(h^m)$; $h$ is the stepsize of the finite-difference method. In

*Corresponding author. E-mail: d2846@beta.ist.utl.pt and d3151@beta.ist.utl.pt.

0377-0427/95/$09.50 © 1995 Elsevier Science B.V. All rights reserved

SSDI 0377-0427(94)00065-4
other extrapolation methods for ordinary differential equations, the validity of an asymptotic expansion of the same kind as (1.1) is also required. For example, the method developed in [12] is based on an asymptotic expansion in even powers of $h$. A correlated approach is the defect correction method introduced in [28]. It consists essentially in adding to the original numerical solution corrections that eliminate the successive terms of the expansion, thus obtaining more and more accurate results.

An asymptotic expansion like (1.1) will be valid if the coefficients and the right-hand side of the differential equation are sufficiently smooth functions and the finite-difference scheme is stable. In the case of the boundary value problem

$$ \frac{d}{dx} \left( \frac{du}{dx} \right) - qu = -f, \quad (1.2) $$

$$ u(0) = u(1) = 0, \quad (1.3) $$

the conditions under which the Richardson extrapolation is applicable are discussed in a recent paper [16]. In that paper it was shown that Richardson extrapolation may be applied when the function $f$ has certain kinds of singularities, the conditions of applicability being weaker than it was earlier assumed.

However, it was also pointed out in that work that in the case of many boundary-value problems with singularities an asymptotic error expansion of the form (1.1) is not valid. In such cases, of course, the Richardson extrapolation will not be useful and the way to improve the convergence is to look for an asymptotic error expansion with a form different from (1.1). Such an approach was developed in [22], where this author considered the boundary value problem:

$$ xy'' - y' - x = 0, \quad (1.4) $$

$$ y(0) = y(1) = 0. \quad (1.5) $$

(Here the exact solution is $y(x) = (x^2/2) \log x$ having a singularity at $x = 0$.) Mayers has shown that in this case the asymptotic expansion for the error of the finite-difference method has the form

$$ y^h(x) - y(x) \sim C_1(x) h^2 \log h + C_2(x) h^2 \quad (1.6) $$

where $C_1$ and $C_2$ are functions, independent of $h$. Therefore, in order to accelerate the convergence in this case, the author recommends to use an extrapolation algorithm which takes into account (1.6). The numerical scheme developed in this work is based on the same idea, although the asymptotic error expansions are obtained by a different method.

It is worth here to refer the results obtained in the closely related field of numerical integration. In this domain the Romberg integration [4–6, 13, 14, 25] is a well known extrapolation method analogous to Richardson extrapolation. In the Romberg method the integration error must be representable as a power series of the stepsize $h$. If such an expansion is not available, alternative methods were analyzed, such as the $e$-algorithm [1, 4, 5]. Genz [9] reports that this algorithm is successfully applied to the integration of functions with endpoint singularities. An important progress in the generalization of Romberg's integration was obtained in [23, 17]. The main results in this field were reviewed in [18]. If the integrand function has one of the forms (a) $f(x) = g(x)x^\alpha(1 - x)^\beta$ or (b) $f(x) = g(x)x^\alpha(1 - x)^\beta \ln x$, where $g(x)$ is analytic and $\alpha, \beta > -1$, the
asymptotic error expansion will be

$$I^h - I \sim \sum_{k=0}^{N-1} A_{k+1} h^{k+1} + \sum_{k=0}^{N-1} C_{k+1} h^{k+1} + O(h^N)$$

(1.7)

or

$$I^h - I \sim \sum_{k=0}^{N-1} A_{k+1} h^{k+1} + \ln h \sum_{k=0}^{N-1} A_{k+1} h^{k+1} + \sum_{k=0}^{N-1} C_{k+1} h^{k+1} + O(h^N),$$

(1.8)

respectively. In the last formulae $I = \int_0^1 f(x)dx$ and $I^h$ is the approximate value of the integral obtained by application of the composite trapezoidal rule with stepsize $h$.

Fox [8] and Shanks [26] showed how the asymptotic expansions of the kind (1.7) may be used to obtain effective extrapolation methods for the integration of singular functions. The most general algorithm of this kind, however, seems to be the E-algorithm of [1, 15]. Sidi [27] uses also a particular case of the E-algorithm under the designation of generalized Richardson extrapolation. More recently, Ford and Sidi [7] have developed an alternative recursive algorithm that computes the E-transform more efficiently.

In the present work we analyse the use of the E-algorithm to improve the accuracy of the solution of boundary-value problems of the kind (1.2)–(1.3). We are particularly interested in the case when the right-hand side function $f$ has different kinds of singularities. Asymptotic expansions of the error, based on the analogous formulae known for the numerical integration, are obtained in Section 2. In Section 3 we analyse some numerical examples and discuss the efficiency of the E-algorithm for these cases.

### 2. Asymptotic error expansions

#### 2.1. Summary of previous results

We shall consider a boundary-value problem for a second-order linear differential equation. This problem is well known and was discussed in a previous paper [16]. It may be formulated as follows:

$$Lu(x) = \frac{d}{dx} \left( p(x) \frac{du}{dx} \right) - q(x)u(x) = -f(x), \quad x \in ]0, 1[ \setminus Y,$$

(2.1)

$$u(0) = u(1) = 0,$$

(2.2)

where $Y$ is a finite set of numbers, such that $Y \subset ]0, 1[$.

We shall now recall some definitions that we introduced in [16].

**Definition 1.** Let $A = [a, b], ]a, b[, [a, b], [a, b[,$ or $]a, b[,$ and let $Y = \{y_1, y_2, \ldots, y_N\}$ be a finite set of numbers, such that $Y \subset A \subset \mathbb{R}$. For each $m \in \mathbb{N}_0$ let $F^m(A, Y)$ denote the class of real functions $f$ such that:

1. $f$ has piecewise continuous derivatives of degree $\leq m$ on $A \setminus Y$. 

(2) For each \( y_v \in Y \) the left and right side limits of \( f^{(k)} \), for \( k = 0, \ldots, m \) exist and are finite when \( x \to y_v \):

\[
\lim_{x \to y_v^-} f^{(k)}(x) = f^{(k)}_-(y_v), \quad \lim_{x \to y_v^+} f^{(k)}(x) = f^{(k)}_+(y_v), \quad k = 0, \ldots, m.
\]

For \( f^{(k)}_-(y_v) \neq f^{(k)}_+(y_v) \), we define

\[
f^{(k)}(y_v) = \frac{1}{2}(f^{(k)}_-(y_v) + f^{(k)}_+(y_v)), \quad k = 0, \ldots, m. \tag{2.3}
\]

**Definition 2.** Let \( A = [a, b] \), \( A = [a, b[ \) or \( A = ]a, b[ \). We shall denote by \( H^m(A, Y) \) \( (m = 0, 1, \ldots) \) the class of functions \( f \) such that

\[
f \in H^m(A, Y), \quad \int_a^b \int_c^x |f^{(m)}(t)| \, dt \, dx < \infty, \quad \forall c \in ]a, b[. \tag{2.4}
\]

Let us assume that the coefficients of Eq. (2.1), for some \( m \geq 0 \) satisfy the conditions

\[
p \in F^{2m+1}([0, 1], Y), \quad p(x) \geq c_1 > 0, \tag{2.5}
\]

\[
q \in F^{2m}([0, 1], Y), \quad q(x) > 0, \quad \forall x \in [0, 1]. \tag{2.6}
\]

In [16] it was shown that if the conditions (2.5) and (2.6) are satisfied, for every function \( f \in H^{2k}([0, 1], Y) \), there is a unique solution \( u(x) \) of the boundary-value problem (2.1)–(2.2) belonging to \( H^{2k+2}([0, 1], Y) \) (with \( k = 0, 1, \ldots, m \)). This solution may be represented by means of the Green function \( G(x, y) \):

\[
u(x) = \int_0^1 G(x, y) f(y) \, dy. \tag{2.7}
\]

In our case the Green function can be written in the form:

\[
G(x, y) = \begin{cases} 
\frac{\alpha(x) \beta(y)}{\alpha(1)}, & \text{if } x \leq y, \\
\frac{\alpha(y) \beta(x)}{\alpha(1)}, & \text{if } x > y,
\end{cases} \tag{2.8}
\]

where \( 0 \leq x, y \leq 1 \), \( \alpha \) and \( \beta \) are solutions of the following Cauchy problems:

\[
L \alpha(x) = 0, \quad \alpha(0) = 0, \quad M \alpha(0) = 1, \tag{2.9}
\]

\[
L \beta(x) = 0, \quad \beta(1) = 0, \quad M \beta(1) = -1, \tag{2.10}
\]

where \( M \) is a linear operator defined by

\[
Mu(x) = p(x) u'(x). \tag{2.11}
\]
We shall consider the discretization of the boundary-value problem (2.1)-(2.2) by the following finite-difference scheme:

\[(L_h u^h)_i = \frac{1}{h^2} (\bar{p}_{i+1} (u^h_{i+1} - u^h_i) - \bar{p}_i (u^h_i - u^h_{i-1})) - q_i u^h_i = -f_i, \quad i = 1, 2, \ldots, n - 1, \quad (2.12)\]

\[u^h_0 = u^h_n = 0, \quad (2.13)\]

where \(\bar{p}_i = p(x_i - h/2), q_i = q(x_i)\) and \(f_i = f(x_i); x_i = ih\) are the knots of a grid \(X_h\) with stepsize \(h\): \(X_h = \{x_i\}_{i=0,1,\ldots,n}\). Let us suppose that all the elements of \(Y\) (points of discontinuity) are grid points. We shall call grid function to \(u^h\), a function defined at all points of \(X_h\) so that \(u^h = u^h(x_i)\) in the case of one variable, and \(u^h_{i,j} = u^h(x_{i,j}), i, j = 0, \ldots, n\) for two variables. We should remember here that the values of \(f, p\) and \(q\) at the grid points are defined as the arithmetic means of the left and right-hand side limits (see Definition 1). The solution of the scheme (2.12)-(2.13) may be obtained using the Green grid function \(G_{ij}^h\) and the formula

\[u^h_i = h \sum_{j=1}^{n-1} G_{ij}^h f_j \quad (2.14)\]

where the Green grid function is given by

\[G_{ij}^h = \begin{cases} \frac{\alpha_i^h \beta_j^h}{\alpha_n^h}, & \text{if } i \leq j, \\ \frac{\alpha_i^h \beta_j^h}{\alpha_n^h}, & \text{if } i > j, \end{cases} \quad (2.15)\]

\(\alpha^h\) and \(\beta^h\) being the solutions of the finite-difference equations which approximate (2.10) and (2.12):

\[(L_h \alpha^h)_i = 0, \quad i = 1, 2, \ldots, n - 1, \quad \alpha_0^h = 0, \quad (M_h \alpha^h)_0 = 1, \quad (2.16)\]

\[(L_h \beta^h)_i = 0, \quad i = 1, 2, \ldots, n - 1, \quad \beta_n^h = 0, \quad (M_h \beta^h)_{n-1} = -1, \quad (2.17)\]

where \(M_h\) is an operator defined by

\[(M_h u^h)_i = \bar{p}_{i+1} \frac{u^h_{i+1} - u^h_i}{h}. \quad (2.18)\]

2.2. Asymptotic expansion of the Green grid function

In order to obtain an asymptotic error expansion, we shall compare the expression of the exact solution, given by (2.7), with the expression of \(u^h\) given by (2.14). With this purpose we need at first to analyse the difference between the exact and the grid Green functions.

**Lemma 3.** Let \(G(x_i, x_j)\) be the Green function for the problem (2.1)-(2.2) and \(G_{ij}^h\) the corresponding grid function, defined above. Then if the functions \(p\) and \(q\) satisfy the prescribed conditions the
following equality holds true:

\[ G_{ij}^h - G(x_i, x_j) = \sum_{k=1}^{m-1} \gamma_{2k}(x_i, x_j) h^{2k} + r_{2m}(x_i, x_j), \]  

(2.19)

where \( \gamma_{2k}(x, y) \) are coefficients independent of \( h \) belonging to \( F^{2m+2-k}([0, 1], Y \cup \{x\}) \) as functions of the second variable, when the first variable is fixed.

\[ \|r_{2m}\| = \max_{1 \leq i,j \leq n} |r_{2m}(x_i, x_j)| \leq C h^{2m}. \]

**Proof.** Let us first consider the difference \( \alpha^h - \alpha \), where \( \alpha \) is the solution of the initial-value problem (2.9) and \( \alpha^h \) is the solution of the finite-difference scheme (2.16). Taking into account the conditions (2.5) and (2.6), it is easy to verify that the local discretization error of the operator \( L \) is

\[ (L_h[\alpha]^h)_i = (L\alpha)(x_i) = \sum_{k=1}^{m-1} l_{2k}(x_i) h^{2k} + \lambda_{2m}(x_i), \quad \forall x_i \in X_h \setminus Y, \]  

(2.20)

where the symbol \([\alpha]^h\) must be understood as the grid function, resulting from the evaluation of \( \alpha \) at the gridpoints, and \( \|\lambda_{2m}\| = O(h^{2m}) \). On the other hand, the local discretization error of the operator \( M \), defined by (2.11), is given by

\[ (M_h[\alpha]^h)_0 - M\alpha(0) = p_1 \frac{\alpha(x_1) - \alpha(x_0)}{h} - \alpha'(0) p(0) = \sum_{k=1}^{2m-1} \mu_k h^k + \mu_{2m}, \]  

(2.21)

where \( \mu_{2m} = O(h^{2m}) \).

It may be shown that although the sum in (2.21) contains odd powers of \( h \), the expansion of \( \alpha^h - \alpha \) will contain only even powers. The scheme of the proof is the same as the one used in the expansion theorem [19, pp. 273–279; 20, pp. 16–22]. First we formally represent the error as a series:

\[ \alpha_i^h = \alpha(x_i) + \sum_{j=1}^{m-1} a_{2j}(x_i) h^{2j} + s_{2m}(x_i), \quad \forall x_i \in X_h \setminus Y. \]  

(2.22)

Let us now consider (2.22) as an identity between grid functions and apply the operator \( L_h \) to both of them. Observing that the quantities \( L_h[a_{2j}]^h \) may be expanded in Taylor series, as well as \( L_h[\alpha]^h \) in (2.20) and denoting the coefficients of such expansions by \( l_{2k, 2j} \), we shall obtain

\[ 0 = (L\alpha)(x_i) + h^2((La_2)(x_i) + l_2(x_i)) + \cdots + h^{2j}(La_{2j})(x_i) + l_{2j}(x_i) + \sum_{k=1}^{j-1} l_{2k, 2j-2k}(x_i) \]  

\[ + \cdots + (L_h s_{2m})_i + \rho_h(x_i), \]  

(2.23)

where \( \|\rho_h\| = O(h^{2m}) \).
If we apply now the operator $M_h$ to both sides of (2.22) and carry out the same transformations as done in the case of $L_h$, we shall obtain

$$1 = Mx + hM_1 + h^2(Ma_2 + M_2) + \cdots + h^{2j} \left( Ma_{2j} + M_{2j} + \sum_{k=1}^{j-1} M_{2k,2j-2k} \right) + h^{2j+1} \left( M_{2j+1} + \sum_{k=0}^{j-1} M_{2k+1,2j-2k} \right) + \cdots + M_h s_{2m} + \sigma_h,$$

(2.24)

where $||\sigma_h|| = O(h^{2m})$, $M_k$ are the coefficients of the series in the right-hand side of (2.21) and the symbols $M_{k,j}$ denote the coefficients of the Taylor expansion of $M_h a_{2j}$.

By equating to 0 the coefficients of the even powers of $h$, up to $2m - 2$, in both the series (2.23) and (2.24) we obtain the initial-value problems

$$(La_{2j})(x) = - l_{2j}(x) - \sum_{k=1}^{j-1} l_{2k,2j-2k}(x), \quad x \in \mathbb{R} \setminus \{ \mathbb{Y} \},$$

(2.25)

$$a_{2j}(0) = 0,$$

$$M a_{2j} = - \sum_{k=1}^{j-1} M_{2k,2j-2k}, \quad j = 1, \ldots, m - 1.$$  

(2.26)

Hence, if the $a_{2j}$ are defined as the solutions of (2.25) and (2.26) all the terms corresponding to even powers in (2.23) and (2.24) vanish, up to $2m - 2$. Moreover, comparing the expressions of the coefficients $l_{2k,2j}(0)$ and $M_{2k,j}$, we may easily verify that the odd powers in the right-hand side of (2.24) also vanish, up to $2m - 1$, for the given choice of the $a_{2j}$. Therefore the identities (2.23) and (2.22) become

$$(L_h s_{2m})_i = - \rho_h(x_i), \quad x_i \in X_h \setminus \mathbb{Y},$$

(2.27)

$$(M_h s_{2m})_0 = - \sigma_h,$$

(2.28)

where $||\rho_h|| = O(h^{2m})$ and $\sigma_h = O(h^{2m})$. Moreover, we have obviously $s_{2m}(0) = 0$. From the well-known stability of the finite-difference scheme it follows that there is an unique grid function $s_{2m}$ satisfying this equations and $||s_{2m}|| = O(h^{2m})$.

So we have proved that expansion (2.22) for the function $x^k$ is valid for all $x_i \in X_h \setminus \mathbb{Y}$. Using the same argumentation, we can show that

$$\beta^j_i = \beta(x_j) + \sum_{k=1}^{m-1} b_{2k}(x_j) h^{2k} + t_{2m}(x_j),$$

(2.29)

where $\beta^j_i$ is the solution of the finite-difference scheme (2.17) and $\beta(x)$ is the solution of the initial-value problem (2.17); $b_{2k}(x)$ are functions independent of $h$; $b_{2k} \in F^{2m+2-2k}([0, 1], \mathbb{Y})$ and

$$||t_{2m}|| = \max_{1 \leq i \leq n} |t_{2m}(x_i)| = O(h^{2m}).$$

(2.30)
Furthermore from (2.22) it follows in particular that the quantity $\alpha_n^h$, figuring in the expression of the Green grid function, may be expanded in powers of $h$. Therefore its reciprocal $1/\alpha_n^h$ may be itself expanded as follows:

$$
\frac{1}{\alpha_n^h} = \frac{1}{\alpha(1)} \frac{1}{1 + \sum_{k=1}^{m-1} \tilde{a}_{2k}(1) \frac{1}{\alpha(1)} h^k + O(h^{2m})} = \frac{1}{\alpha(1)} + \sum_{k=1}^{m-1} \tilde{a}_{2k} h^{2k} + O(h^{2m}),
$$

(2.31)

where the $\tilde{a}_{2k}$ coefficients are independent of $h$. (The validity of such an expansion follows from the fact that the sum in the denominator approaches 1, as $h \to 0$.)

Finally, the validity of the expansion (2.19) follows from the equalities (2.22), (2.29) and (2.31).

2.3. Asymptotic expansion of $u^h - u$

In [16] it was established that the scheme (2.12)–(2.13) has a unique solution $u^h$ if the function $j$ belongs at least to $H^9([0, 1[, \ Y)$; moreover, by the Theorem 1 in [16], if $f \in H^{2m}([0, 1[, \ Y)$, with $m \geq 1$, the following asymptotic error expansion is valid:

$$
u(x_i) = u_i^h + \sum_{k=1}^{m-1} \tau_{2k}(x_i) h^{2k} + \rho_{2m}(x_i), \quad i = 1, 2, \ldots, n - 1,
$$

(2.32)

where $\|\rho_{2m}\|_{2h} = \max_{1 \leq i \leq n} |\rho_{2m}(x_i)| \leq C h^{2m}$, for some positive constant $C$, when $h \to 0$. The functions $\tau_{2k}$ are independent of $h$ and belong to $H^{2m+2-2k}([0, 1[, \ Y)$, $k = 1, \ldots, m - 1$.

Our purpose in the present paper is to obtain an asymptotic expansion with more terms than (2.32) for the particular case where the right-hand side function has one of the forms

$$
 f(x) = g(x) x^\mu (1 - x)^\nu
$$

(2.33)

or

$$
 f(x) = g(x) x^\mu (1 - x)^\nu \ln x,
$$

(2.34)

where $g \in F^{2m}([0, 1[, \ Y, \ \mu > -2, \ \nu > -2$. As we pointed out in [16], under these conditions, functions (2.33) and (2.34) belong at least to $H^9([0, 1[, \ Y)$ and therefore the boundary-value problem as well as the finite-difference scheme has a unique solution. However when $\mu$ or $\nu$ are negative, $f \not\in H^2([0, 1[, \ Y)$ and the expansion (2.32) does not make sense.

Using the asymptotic expansion of the Green grid function obtained above and the generalization of the Euler–Maclaurin formula for the case of singular integrands, we shall now derive an asymptotic expansion of the error of the finite-difference solution.

**Theorem 4.** Suppose that the coefficients of Eq. (2.1) satisfy the conditions (2.5) and (2.6) and all the elements of $\ Y$ are gridpoints. Let the right-hand side function have one of the forms (2.33) or (2.34), with $\mu > -2$ and $\nu > -2$. Then the following asymptotic error expansions are valid.

In the case of (2.33)

$$
u(x_i) = u_i^h + \sum_{k=1}^{m-1} A_{2k}(x_i) h^{2k} + \sum_{k=1}^{[2m-\mu-1]} B_k(x_i) h^{k+\mu+1} + \sum_{k=1}^{[2m-\nu-1]} D_k(x_i) h^{k+\nu+1} + \rho_{2m}(x_i).
$$

(2.35)
In the case of (2.34)

\[ u(x_i) = u_i^h + \sum_{k=1}^{m-1} A_{2k}(x_i) h^{2k} + \sum_{k=1}^{[2m-\mu-1]} h^{k+\mu+1} (B_k(x) + B_k(x) \ln h) \]

\[ + \sum_{k=1}^{[2m-\nu-1]} h^{k+\nu+1} (D_k(x_i) + D_k(x_i) \ln h) + \rho_{2m}(x_i), \quad i = 1, 2, \ldots, n - 1, \]  

(2.36)

where \( \|\rho_{2m}\| = O(h^{2m}) \), when \( h \to 0 \) and \( A_{2k}, B_k, B_k', D_k, D_k' \) are functions independent of \( h \). (The symbol \( [\sigma] \) stands for the integer part of \( \sigma, \forall \sigma \in \mathbb{R} \).)

**Proof.** From (2.7) and (2.14) we obtain

\[ u(x_i) - u_i^h = \Delta^1_i(x_i) + \Delta^2_i(x_i), \]  

(2.37)

where

\[ \Delta^1_i(x_i) = \int_0^1 G(x_i, t) f(t) dt - h \sum_{j=1}^{n-1} G(x_i, x_j) f(x_j) \]  

(2.38)

and

\[ \Delta^2_i(x_i) = h \sum_{j=0}^n [G(x_i, x_j) - G^h](x_j). \]  

(2.39)

Using (2.8) we obtain

\[ \int_0^1 G(x, t) f(t) dt = \frac{1}{\alpha(1)} \left( \beta(x) \int_0^x \alpha(t) f(t) dt + \alpha(x) \int_x^1 \beta(t) f(t) dt \right). \]  

(2.40)

Analogously, we have

\[ h \sum_{j=1}^{n-1} G(x_i, x_j) f(x_j) = h \left( \frac{\beta(x_i)}{\alpha(1)} \sum_{j=1}^{i-1} \alpha(x_j) f(x_j) + \alpha(x_i) \sum_{j=i}^{n-1} \beta(x_j) f(x_j) \right). \]  

(2.41)

If \( f \) has the form (2.33), we may write

\[ \alpha(x) f(x) = \frac{\alpha(x)}{x} \chi^{\mu+1}(1 - x)^v g(x) \]  

(2.42)

where \( \alpha(x)/x \in F^{2m+1}([0, 1], Y) \), since \( \alpha \in F^{2m+2}([0, 1], Y) \) and \( \alpha(0) = 0 \). Moreover, since \( \mu > -2 \), the product \( \alpha(x) f(x) \) is integrable in \([0, x]\) (with \( 0 < x < 1 \)). Moreover, using the generalized Euler–Maclaurin expansion (1.7) we may write

\[ h \sum_{j=1}^{i-1} \alpha(x_j) f(x_j) + \frac{h}{2} \alpha(x_i) f(x_i) - \int_0^{x_i} \alpha(t) f(t) dt \]

\[ = \sum_{k=1}^{m} A_{2k}(x_i) h^{2k} + \sum_{k=1}^{[2m-\mu-1]} B_k(x_i) h^{k+\mu+1} + r_{2m}(x_i), \]  

(2.43)

where \( \|r_{2m}\| = o(h^{2m}). \)
In the same way, we have

$$\beta(x)f(x) = \frac{\beta(x)}{1 - x}x^\mu (1 - x)^{\nu + 1}g(x),$$

(2.44)

where $\beta(x)/(1 - x) \in F^{2m+1}(0, 1, Y)$, since $\beta \in F^{2m+2}(0, 1, Y)$ and $\beta(1) = 0$. Since $\nu > -2$, $\beta(x)f(x)$ is integrable in $[x, 1]$, with $0 < x < 1$. Therefore, using the generalized Euler–Maclaurin expansion (1.7), we shall have

$$h \sum_{j=1}^{n-1} \alpha(x_j)f(x_j) + \frac{h}{2} \beta(x_i)f(x_i) - \int_{x_i}^{1} \beta(t)f(t) \, dt = \sum_{k=1}^{m} A_{2k}(x_i)h^{2k} + \sum_{k=1}^{[2m-\nu-1]} B_k(x_i)h^{k+\nu+1} + r_{2m}'(x_i),$$

(2.45)

where $|r_{2m}'| = O(h^{2m})$.

From (2.38), (2.43) and (2.45) we obtain

$$A_{1}^{k} = \sum_{k=1}^{m-1} A_{2k}(x_i)h^{2k} + \sum_{k=1}^{[2m-\mu-1]} B_k(x_i)h^{k+\mu+1} + \sum_{k=1}^{[2m-\nu-1]} B_k(x_i)h^{k+\nu+1} + \rho_{2m}(x_i),$$

(2.46)

where $B_k$ and $\bar{B}_k$ are functions of the class $F^{2m}(0, 1, Y)$ and $A_{2k}$ has the form

$$A_{2k}(x) = a_{2k}(x)x^{\mu+2-2k}$$

(2.47)

with $a_{2k}$ belonging to $F^{2m}(0, 1, Y)$.

Consider now the difference $A_{2}^{k}(x)$, given by (2.39). Using Lemma 3, we have

$$A_{2}^{k}(x_i) = - \sum_{k=1}^{m} \left( h \sum_{j=1}^{n-1} \gamma_{2k}(x_i, x_j)f(x_j) \right)h^{2k} + v_{2m}(x_i)$$

(2.48)

where $|v_{2m}| = O(h^{2m})$.

Moreover Lemma 3 guaranteed that $\gamma_{2k}(x_i, x_j) \in F^{2m+2-k}(0, 1, Y \cup \{x_i\})$ with respect to the second variable, $\gamma_{2k}(x_i, 0) = 0$ and $\gamma_{2k}(x_i, 1) = 0 \forall x_i \in [0, 1]$. Therefore, $\gamma_{2k}(x_i, x)$ is integrable in $[0, 1]$ with respect to the second variable. Using the generalized Euler–Maclaurin expansion (1.7), we may write

$$h \sum_{j=0}^{n} \gamma_{2k}(x_i, x_j)f(x_j) = \int_{0}^{1} \gamma_{2k}(x_i, x)f(x) \, dx + \sum_{l=1}^{m-k} G_{2k, 2l}(x_i)h^{2l} + \sum_{k=1}^{[2m-\mu-1]} B_{2k, l+\mu+1}(x_i)h^{l+\mu+1}$$

$$+ \sum_{k=1}^{[2m-\nu-1]} D_{2k, l+\nu+1}(x_i)h^{l+\nu+1} + t_{2m-2k}(x_i), \quad k = 1, 2, \ldots, m,$$

(2.49)

where $G(x)$, $B(x)$ and $D(x)$ are coefficients independent of $h$, and $|t_{2m-2k}| = O(h^{2m-2k})$. From (2.48) and (2.49) we obtain

$$A_{2}^{k}(x_i) = \sum_{k=1}^{m} A_{2k}(x_i)h^{2k} + \sum_{k=1}^{[2m-\mu-1]} B_k(x_i)h^{k+\mu+1} + \sum_{k=1}^{[2m-\nu-1]} D_k(x_i)h^{k+\nu+1} + w_{2m}(x_i),$$

(2.50)
where

\[
\mathcal{I}_{2k}(x_i) = \int_0^1 \gamma_{2k}(x_i, x) f(x) \, dx + \sum_{i=1}^{k-1} c_{2i, 2k-2i}(x_i)
\]  

(2.51)

with \( k = 1, 2, \ldots, m \) and \( l = 1, 2, \ldots, k - 1; \|w_{2m}\| = O(h^{2m}) \). Finally, from (2.50), (2.46) and (2.37), we obtain (2.35). If the right-hand side has the form (2.34), the expansion (2.36) may be obtained as we did above for (2.35), but using (1.8) instead of (1.7) to evaluate the integration error.

**Remark.** For some values of \( \mu \) and \( \nu \) the expressions in the right-hand side of (2.35) and (2.36) may be considerably simplified. If \( \mu \) (or \( \nu \)) is a non-negative integer, the second (or the third) sum vanishes in the right-hand side of (2.35).

Moreover, if \( \mu \) and \( \nu \) differ by an integer, the second and the third sums in the right-hand side of (2.35) and (2.36) reduce to a single sum. In the case \( \mu \) and \( \nu \) are both non-negative integers the expression in the right-hand side of (2.35) reduces to a series in even powers of \( h \).

The terms of the asymptotic error expansion, obtained by Theorem 4 for different forms of \( f(x) \), are displayed in Table 1.

### 2.4. Evaluation of integrals of \( u(x) \)

In many practical problems we are concerned about linear functionals of the solution of a certain boundary-value problem. In [16] we introduced for example the integral

\[
\Phi(f) = \int_0^1 f(x) u(x) \, dx,
\]

(2.52)

where \( f \) is the function on the right-hand side of the problem (2.1)-(2.2) and \( u \) is the exact solution of that problem.

Let \( \Phi_h(f) \) be the finite-difference approximation to \( \Phi(f) \), given by the formula

\[
\Phi_h(f) = h \sum_{i=1}^{n-1} f(x_i) u_i^h,
\]

(2.53)

where \( u_i^h \) is a component of the solution of the finite-difference scheme (2.12)-(2.13).

### Table 1

<table>
<thead>
<tr>
<th>( f(x) )</th>
<th>Terms of the asymptotic expansion of ( u(x) - u^h(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ln x )</td>
<td>( A_i(x) h^i \ln h, B_i(x) h^i (i = 1, 2, \ldots) )</td>
</tr>
<tr>
<td>( x^k \ln x ) (( k \in \mathbb{N} ))</td>
<td>( A_{2i}(x) h^{2i}, B_i(x) h^{2i+i+1} \ln h, C_i(x) h^{2i+i+1} )</td>
</tr>
<tr>
<td>( x^\mu \ln x ) (( \mu &gt; -2, \mu \neq -1 ))</td>
<td>( A_{2i}(x) h^{2i}, B_i(x) h^{2i+i+1} \ln h, C_i(x) h^{2i+i+1} )</td>
</tr>
<tr>
<td>( 1/x ) or ( 1/\sin x )</td>
<td>( A_i(x) h^i )</td>
</tr>
<tr>
<td>( x^\mu ) (( \mu &gt; -2, \mu \neq -1 ))</td>
<td>( A_{2i}(x) h^{2i}, B_i(x) h^{2i+i+1} )</td>
</tr>
</tbody>
</table>
Theorem 5. Let $f$ have one of the forms (2.33) or (2.34), with $\mu > -\frac{1}{3}$ and $v = 0$. Then the functional $\Phi$ defined by (2.52) exists and $\Phi_h \to \Phi$, as $h \to 0$. Moreover the following asymptotic expansions are valid:

(a) If $f$ has the form (2.33), with $\mu > -\frac{1}{3}$, $\mu \neq -1$, $v = 0$, then

$$\Phi(f) - \Phi_h(f) = \sum_{k=1}^{m-1} A_{2k} h^{2k} + \sum_{k=1}^{[2m - 2 - 2\mu]} C_k h^{k+\mu+1} + \sum_{k=1}^{[2m - 1 - \mu]} B_k h^{k+2\mu+2} + O(h^{2m}).$$

(b) If $f$ has the form (2.33), with $\mu = -1$, $v = 0$, then

$$\Phi(f) - \Phi_h(f) = \sum_{k=1}^{2m-1} h^k (A'_k \ln h + A_k) + O(h^{2m}).$$

(c) If $f$ has the form (2.34), with $\mu > -\frac{1}{3}$, $\mu \neq -1$, $v = 0$, then

$$\Phi(f) - \Phi_h(f) = \sum_{k=1}^{m-1} A_{2k} h^{2k} + \sum_{k=1}^{[2m - 1 - \mu]} C_k h^{k+\mu+1} (C'_k \ln h + C'_k)$$

$$+ \sum_{k=1}^{[2m - 2 - 2\mu]} h^{k+2\mu+2} (B_k (\ln h)^2 + B'_k \ln h + B'_k) + O(h^{2m}).$$

(d) If $f$ has the form (2.34), with $\mu = -1$, $v = 0$, then

$$\Phi(f) - \Phi_h(f) = \sum_{k=1}^{2m-1} h^k (A''_k (\ln h)^2 + A''_k (\ln h)^2 + A'_k \ln h + A_k) + O(h^{2m}).$$

Proof. We have

$$\Phi(f) - \Phi_h(f) = \int_0^1 u(x) f(x) \, dx - h \sum_{i=1}^{n-1} u_i f(x_i) = \delta_1^h + \delta_2^h,$$

where

$$\delta_1^h = \int_0^1 u(x) f(x) \, dx - h \sum_{i=1}^{n-1} u(x_i) f(x_i)$$

and

$$\delta_2^h = h \sum_{i=1}^{n-1} u(x_i) f(x_i) - h \sum_{i=1}^{n-1} u_i f(x_i) = h \sum_{i=1}^{n-1} (u(x_i) - u_i) f(x_i).$$

Let us first consider the case when $f$ has the form (2.33) with $v = 0$, that is $f(x) = x^u g(x)$, with $\mu > -\frac{1}{3}$ and $g \in F^{2m}([0, 1], \mathbb{Y})$. Using (2.40) to represent the exact solution we obtain

$$\Phi(f) = \int_0^1 f(x) \alpha(t) \int_0^x \beta(t) f(t) \, dt + \alpha(x) \int_x^1 \beta(t) f(t) \, dt \, dx = I_1 + I_2,$$

where

$$I_1 = \frac{1}{\alpha(1)} \int_0^1 f(x) \beta(x) \int_0^x \alpha(t) f(t) \, dt \, dx.$$
and
\[ I_2 = \frac{1}{\alpha(1)} \int_0^1 f(x) \zeta(x) \int_x^1 \beta(t) f(t) \, dt \, dx. \] (2.63)

Let us define
\[ \phi_1(x) = \int_0^x \alpha(t) f(t) \, dt = \int_0^x \alpha(t) t^{\mu} g(t) \, dt. \] (2.64)

Since \( f \) has the form (2.33) and \( \alpha(0) = 0 \), by (2.9), the integral in the right-hand side of (2.64) is convergent \( \forall x \geq 0 \), and the following expression is valid:
\[ \phi_1(x) = x^{\mu+1} \sum_{k=1}^{\infty} a_k x^k, \quad \forall x \geq 0, \] (2.65)

where \( a_k \) are real constants that do not depend on \( x \), \( k = 1, 2, \ldots \).

Hence we may rewrite (2.62) under the form
\[ I_1 = \frac{1}{\alpha(1)} \int_0^1 f(x) \beta(x) \phi_1(x) \, dx = \frac{1}{\alpha(1)} \sum_{k=1}^{\infty} a_k \int_0^1 \beta(x) g(x) x^{2\mu+1+k} \, dx. \] (2.66)

Now if we denote by \( I_1^h \) the approximate value of \( I_1 \) estimated by the trapezoidal rule with stepsize \( h \), we obtain
\[ I_1 - I_1^h = -\frac{1}{\alpha(1)} \sum_{k=1}^{\infty} (I_{1,k}^h - I_{1,k}), \] (2.67)

where \( I_{1,k} \) denotes each integral in the right-hand side of (2.66) and \( I_{1,k}^h \) represents the approximate value of this integral obtained by the trapezoidal rule. Applying the asymptotic expansion (1.7) to each term of the sum (2.67) we obtain
\[ I_1 - I_1^h = \sum_{j=1}^{[2m-2\mu-2]} B_j h^{2\mu+2+j} + \sum_{j=1}^{m} A_{2j} h^{2j} + o(h^{2m}). \] (2.68)

Let us now define
\[ \phi_2(x) = \int_x^1 \beta(t) f(t) \, dt. \] (2.69)

This integral is finite if \( x \neq 0 \). If \( \mu \neq -1 \) the following expansion is valid
\[ \phi_2(x) = b_0 + x^{\mu} \sum_{k=1}^{\infty} b_k x^k \] (2.70)

\( (b_k \) denotes a real constant that does not depend on \( x \), as well as the \( b'_k \) and \( b''_k \) appearing below, for \( k = 0, 1, 2, \ldots \)).
Therefore we may rewrite (2.63) under the form

\[ I_2 = \frac{1}{\alpha(1)} \int_0^1 f(x)\, x(\phi(x)\, dx \]

\[ = \frac{b_0}{\alpha(1)} \int_0^1 g(x)\, x^{\mu+1} \, dx + \frac{1}{\alpha(1)} \sum_{k=1}^\infty b_k \int_0^1 x^\mu \, dx. \]

(2.71)

All the integrals in the right-hand side of (2.71) converge since \( \mu > -\frac{3}{2} \). Thus if we approximate each term of this sum by the trapezoidal rule (as we did with (2.66)) and we apply the asymptotic expansion (1.7) to the error of this approximation, we shall obtain the following asymptotic expansion:

\[ I_2 - I_2^h = \sum_{j=1}^{\left[2m-\mu-2\right]} C_j h^{\mu+2+j} + \sum_{j=1}^{\left[2m-2\mu-2\right]} B_j h^{2\mu+2+j} + \sum_{j=1}^m A_{2j} h^{2j} + o(h^{2m}). \]

(2.72)

From (2.61), (2.68) and (2.72) we obtain

\[ \delta_1^h = I_2 - I_2^h + I_1 - I_1^h = \sum_{j=1}^{\left[2m-\mu-2\right]} C_j h^{\mu+2+j} + \sum_{j=1}^{\left[2m-2\mu-2\right]} B_j h^{2\mu+2+j} \]

\[ + \sum_{j=1}^m A_{2j} h^{2j} + o(h^{2m}). \]

(2.73)

In the case \( \mu = -1 \) Eq. (2.68) remains valid but instead of (2.70) we shall have

\[ \phi_2(x) = \int_x^1 \frac{g(t)}{t} \, dt = \sum_{k=0}^\infty (b_k + b'_k \ln x) x^k. \]

(2.74)

Therefore using (1.8) the asymptotic expansion of \( \delta_1^h \) will become

\[ \delta_1^h = \sum_{k=1}^{2m} h^k (\hat{A}_k + \hat{A}'_k \ln h) + o(h^{2m}). \]

(2.75)

Let us now consider the case when \( f \) has the form (2.34), that is, when \( f(x) = x^\mu \ln x \, g(x) \). In this case, using the same arguments, we can easily verify that if \( \mu \neq -1 \) we have the following expansion for \( \delta_1^h \):

\[ \delta_1^h = h^{\mu+2} (C_1 \ln h + \hat{C}_1') + \sum_{k=1}^{\left[2m-2\mu-2\right]} h^{2\mu+2+k} (\hat{B}_k (\ln h)^2 + \hat{B}'_k \ln h + \hat{B}''_k) \]

\[ + \sum_{k=1}^m \hat{A}_{2k} h^{2k} + o(h^{2m}). \]

(2.76)

On the other hand, if \( \mu = -1 \) we shall have

\[ \phi_2(x) = \int_x^1 \frac{g(t)}{t} \, dt = \sum_{k=0}^\infty (b_k + b'_k \ln x + b''_k (\ln x)^2) x^k. \]

(2.77)
From here, we can easily obtain

\[ \delta_1^h = \sum_{k=1}^{2m} h^k \left( \tilde{A}_k (\ln h)^3 + \tilde{A}_k' (\ln h)^2 + \tilde{A}_k'' \ln h + \tilde{A}_k''' \right) + o(h^{2m}). \quad (2.78) \]

Until now we obtained the asymptotic expansions of \( \delta_1^h \) for the 4 cases, considered in the statement of the theorem. Let us now derive the corresponding expansions of \( \delta_2^h \). By (2.60) and Theorem 4, if \( f \) has the form (2.33) we have

\[ \delta_2^h = \sum_{i=1}^{n-1} (u(x_i) - u_i^h) f(x_i) \]

\[ = \sum_{i=1}^{n-1} \left( \sum_{k=1}^{m-1} A_{2k}(x_i) h^{2k} + \sum_{k=1}^{2m-1} \frac{B_k(x_i)}{k!} h^{k+1} + \frac{\rho_{2m}(x_i)}{k!} \right) f(x_i). \quad (2.79) \]

Since by (2.47) \( A_{2k}(x) = a_{2k}(x) x^{\mu+2-2k} \), where \( a_{2k}(x) \in F^{2m}([0, 1], Y) \), using the Euler–Maclaurin expansion we may write

\[ h \sum_{i=1}^{n-1} A_{2k}(x_i) f(x_i) = \int_0^1 A_{2k}(x) f(x) \, dx \]

\[ + \sum_{j=1}^{m-k-1} \mathcal{J}_{2j, 2k} h^{2j} + \sum_{j=1}^{[2m-2-2\mu-2]} \mathcal{J}_{j, k} h^{2\mu+2+j} + O(h^{2m-2k}). \quad (2.80) \]

From (2.80) we obtain

\[ h^{2k} \left( h \sum_{i=1}^{n-1} A_{2k}(x_i) f(x_i) \right) = h^{2k} \int_0^1 A_{2k}(x) f(x) \, dx \]

\[ + \sum_{j=1}^{m-k-1} \mathcal{J}_{2j, 2k} h^{2k+2j} + \sum_{j=1}^{[2m-2-2\mu-2]} \mathcal{J}_{j, k} h^{2\mu+2+k+j} + O(h^{2m}). \quad (2.81) \]

In the same way, since \( B_k \in F^{2m}([0, 1], Y) \), we may verify that

\[ h^{k+\mu+1} \left( h \sum_{i=1}^{n-1} B_k(x_i) f(x_i) \right) = h^{k+\mu+1} \int_0^1 B_k(x) f(x) \, dx + \sum_{j=1}^{[(2m-2\mu-1)/2]} \mathcal{J}_{j, k} h^{\mu+k+1+j} \]

\[ + \sum_{j=1}^{[2m-2\mu-2]} \mathcal{J}_{j, k} h^{2\mu+2+k+j} + O(h^{2m}). \quad (2.82) \]

From (2.79), (2.81) and (2.82) we obtain

\[ \delta_2^h = \sum_{k=1}^{m} \mathcal{J}_{2k} h^{2k} + \sum_{k=1}^{[(2m-2\mu)/2]} \mathcal{J}_{k} h^{\mu+1+k} + \sum_{k=1}^{[2m-2\mu-2]} \mathcal{J}_{k} h^{2\mu+2+k} + O(h^2m). \quad (2.83) \]
Analogously if \( f \) has the form (2.34), it may be shown that

\[
\delta_2^h = \sum_{k=1}^{m} a_{2k} h^{2k} + \sum_{k=0}^{[\frac{2m-μ-2}{2}]} h^{μ+1+k}(\beta_k + \gamma_k \ln h) + \sum_{k=1}^{[\frac{2m-2-μ-2}{2}]} h^{2μ+2+k}(\beta_k(\ln h)^2 + \gamma_k' \ln h + \beta_k''(\ln h)^2) + O(h^{2m}).
\]

(2.84)

The proof of the theorem may be now concluded: (2.54) is obtained from (2.73) and (2.83); (2.55) is obtained from (2.75) and (2.83); (2.56) is obtained from (2.76) and (2.84); (2.57) is obtained from (2.78) and (2.84).

\( \square \)

**Remark.** Asymptotic error expansions for the case when \( ν \neq 0, ν > -\frac{1}{2} \) may be obtained in the same way. We did not present them because the respective expressions are long and cumbersome.

3. **Application of the E-algorithm to boundary-value problems**

When an expansion of the error is known, such as the ones considered in the previous section, a natural way to accelerate the convergence of the corresponding finite-differences scheme (2.12)-(2.13) is to use the E-algorithm of [2-4, 15]. This is a very general extrapolation algorithm designed exactly under the assumption that we know the asymptotic error expansion for a given sequence \( (S_n) \), that is \( S_n = S + a_1 g_1(n) + a_2 g_2(n) + \cdots \) where \( g_i(n) \) are predefined sequences.

In Section 3.1 a short description of the E-algorithm is given as well as some references are made to a subroutine of [5] for this algorithm. That subroutine, called EALGO, and others were used in a multipurpose FORTRAN77 program we implemented to compare the performances of several extrapolation methods. Our program, called EXTRAP01, incorporates besides EALGO, the subroutines EPSRHO, IDELTA, RICHAR and THETA [5] for the related extrapolation processes \( ε \) (with its first and second generalizations [5]), \( ρ, θ, Δ^2 \)-iterated [5, 29] and Richardson extrapolation [1, 5]. Three curvature-type [10] algorithms have been also included.

Every time we need to use the subroutine EALGO to extrapolate the values of a sequence \( (S_n) \) for which an expansion of the error is known, it is necessary to alter the FORTRAN code defining certain quantities related to the sequences \( g_i(n) \) of the error expansion. Concerning all our problems and for the related error expansions in Tables 1 and 2, that code is straightforward and once established can remain unchanged if we settle a convenient main program, enabling us to deal successively with different functions \( f \). In the Appendix are given the main lines of such a code based on the expressions defining the referred error expansions.

The main known results about convergence and convergence acceleration for the E-algorithm [2, 5, 21] are reviewed in Section 3.2. It is particularly easy to verify them using the sequences generated by application of the finite-difference scheme (2.12)-(2.13). In Section 3.3 some numerical examples illustrate how the convergence is dramatically improved if we use the E-algorithm: (i) we want to compute \( Φ \) (see definition (2.52)) with \( f(x) = x^{-1.4} \) (ii) we need the solution \( u(0.5) \) as in (1.2)-(1.3) with \( f(x) = \ln(x)/x \) (iii) again \( u(0.5) \) but \( f(x) = 1/\sin(x) \) and (iv) the same as (ii) but for the computation of \( Φ \). For all this examples the E-algorithm gives numerical results at least comparable with the best of all the other above referred extrapolation algorithms.
3.1. The E-algorithm

If \( g_i(n), i = 1, \ldots, k \) are known sequences and \( (S_n) \) is a sequence whose error has the form

\[
S_n = S + a_1g_1(n) + a_2g_2(n) + \cdots + a_kg_k(n) \quad n = 0, 1, \ldots
\]

then we can compute \( S \) solving the following linear system (assumed non-singular):

\[
S_{n+i} = S + a_1g_1(n+i) + \cdots + a_kg_k(n+i) \quad i = 0, \ldots, k.
\]

If the terms of \( (S_n) \) do not satisfy (3.1) the value of \( S \) computed by (3.2) will depend on \( n \) and \( k \). It will be denoted by \( E^n_k \), and can be determined by the ratio of determinants

\[
E^n_k = \begin{vmatrix}
S_n & S_{n+1} & \cdots & S_{n+k} \\
g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\
\vdots & \vdots & \ddots & \vdots \\
g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \\
1 & 1 & \cdots & 1 \\
g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\
\vdots & \vdots & \ddots & \vdots \\
g_k(n) & g_k(n+1) & \cdots & g_k(n+k)
\end{vmatrix}.
\]

The E-algorithm is a recursive scheme to compute \( E^n_k \) in (3.3) avoiding the computation of determinants, exactly in the same way as the well-known \( \varepsilon \)-algorithm \([1, 5]\) is a scheme to compute (3.3) when the \( g_i \)'s are \( g_i(n) = \Delta S_{n+i-1}, i = 1, 2, \ldots \).

The formulas for this algorithms are \([3]\):

Initializations:

\[
E_0^n = S_n, \quad n = 0, 1, \ldots
\]

\[
g_0^{(n)} = g_i(n), \quad i = 1, 2, \ldots, n = 0, 1, \ldots
\]
For $k = 1, 2, \ldots$; for $n = 0, 1, \ldots$

\begin{equation}
E^{(n)}_k = E^{(n)}_{k-1} + g^{(n)}_{k-1,k} \frac{E^{(n+1)}_{k-1} - E^{(n+1)}_{k-1}}{g^{(n)}_{k-1,k} - g^{(n)}_{k-1,k}},
\end{equation}

\begin{equation}
g^{(n)}_{k,i} = g^{(n)}_{k-1,i} + g^{(n)}_{k-1,k} \frac{g^{(n)}_{k-1,i} - g^{(n+1)}_{k-1,i}}{g^{(n)}_{k-1,k} - g^{(n)}_{k-1,k}}, \quad i = k + 1, k + 2, \ldots.
\end{equation}

The quantities $E^{(n)}_k$ and $g^{(n)}_{k,i}$ are usually placed in a double-entry array, the so-called E-array and $g_i$ array as follows:

\begin{align*}
E^{(0)}_0 &= S_0 \\
E^{(1)}_0 &= S_1 \\
E^{(2)}_0 &= S_2 \\
E^{(3)}_0 &= S_3 \\
&\vdots \\
E^{(5)}_0 &= S_5 \\
&\vdots \\
&\vdots
\end{align*}

\begin{equation}
(3.5)
\end{equation}

In [2, 3] we can find a detailed explanation about these arrays, as well as for the respective programming given in the subroutine EALGO [5]. We incorporated EALGO and other scalar extrapolation subroutines of [5] in a FORTRAN77 program able to compare the performances of such extrapolation processes namely using sequences generated with our difference schemes. The code for the formulas defining $g^{(n)}_{k,i}$ must be given in the calling program of EALGO subroutine. Evidently such formulas concerning our problems will be based on the expansions given in Tables 1 and 2.

In all the examples we considered that the $j$th term of the asymptotic error expansion has the form

\begin{equation}
g_j(n) = h_n^{\alpha_j} \cdot (\ln h_n)^{\beta_j}, \quad j = 1, 2, \ldots, \quad \alpha_j \in \mathbb{R}_+, \quad \beta_j \in \mathbb{N}_0.
\end{equation}

Therefore the expressions for $g_i(k - 1)$ and $g_{k-1}(i - 1)$ (in the notation used in EALGO subroutine) are coded in the calling program of EALGO as follows:

\begin{equation}
h_{k-1} = \frac{1}{n_0 \cdot 2^{k-1}}, \quad r_{k-1} = \ln(h_{k-1}), \quad k = 1, 2, \ldots,
\end{equation}

\begin{equation}
g_i(k - 1) = h_{k-1}^{\alpha_i} \cdot r_{k-1}^{\beta_i}, \quad i = 1, 2, \ldots.
\end{equation}

$(g_{k-1}(i - 1)$ is obtained like $g_i(k - 1)$ by a convenient interchange of indexes — compare with the FORTRAN code in the Appendix.)

Due to computer memory limitations we used the E-algorithm with sequences $(S_n)$ of 12 terms generated by the finite-differences schemes. Thus on the above expressions we took $n_0 = 2$ and $h_{k-1}$ varying from $\frac{1}{2}$ to $(\frac{1}{2})^{12}$. So, for the next examples in Section 3.3, each term of the respective sequence $(S_n)$ is computed as the solution of each tridiagonal linear system of 2 to $2^{12} = 4096$ equations (easily solved by forward and backward substitution) resulting from the finite-differences scheme (2.12)–(2.13).
3.2. Convergence and acceleration results

Comparing with other extrapolation algorithms, relatively few results about convergence and convergence acceleration of the E-algorithm are actually known. Nevertheless concerning our problem almost all the results enunciated below can be easily verified and so we can say that the numerical methods for differential equations with singularities are a particularly interesting field for the application of the E-algorithm. See [4] for a survey of other applications of the E-algorithm.

The next theorem gives sufficient conditions for the convergence of the columns of the E-array.

**Theorem 6** (Brezinski [1, Theorem 5]). Let \( S_n = S + a_1g_1(n) + a_2g_2(n) + \cdots \ \forall n \)

(H1) If \( (S_n) \to S \)

(H2) If \( g_i(n+1)/g_i(n) \to b_i \neq 1 \)

(H3) If \( b_i \neq b_j, i \neq j, i = 1, 2, \ldots \)

then \( E_n^{(k)} \to S, k = 1, 2, \ldots \).

**Remark.** For the \( j \)th term (3.6) of the asymptotic expansion of the error of \( (S_n) \) and with \( h_n \) given by (3.7)

\[
b_i = (1/2)^{a_i}, \ \forall \beta_i
\]

(3.8)

since

\[
\lim_{n \to \infty} \frac{g_i(n+1)}{g_i(n)} = \left(\frac{1}{2}\right)^{a_i} \cdot \lim_{n \to \infty} \left[1 + \frac{\ln(1/2)}{\ln(h_n)}\right]^{\beta_i}.
\]

(3.9)

If besides the conditions (H1), (H2) and (H3), the family of sequences \( g_1(n), g_2(n), \ldots \) form an asymptotic sequence for the error \( S_n - S \) (that is \( g_{i+1}(n) = o(g_i(n)) \)), then every column of the E-array converges faster than the preceding one, as affirms the following:

**Theorem 7** (Brezinski [1, Theorem 7]). Let \( S_n = S + a_1g_1(n) + a_2g_2(n) + \cdots \ \forall n \).

If (H1), (H2), (H3) and

(H4) \( g_{i+1}(n)/g_i(n) \to 0, \ i = 1, 2, \ldots \)

then

\[
(E_{k+1}^{(n)} - S)/(E_k^{(n)} - S) \to 0, \ k = 0, 1, \ldots
\]

(3.10)

**Remark.** Let \( h_n = 1/2^{a_1 + 1}, n = 0, 1, \ldots \) and \( g_i(n) = h_n^{\alpha_i} \) : \( 0 < \alpha_1 < \alpha_2 < \cdots \) then all the conditions in the previous two theorems are satisfied (see Example 9 in Section 3.3).
A common situation is when (H3) fails. Let us take for instance the asymptotic expansion of the error \(\Phi - \Phi_h\) in Table 2 for \(f(x) = \ln(x)/x\) with \(h_n = 1/(n_02^n)\) and \(r_n = \ln(h_n)\):

\[
\Phi_h = \Phi + A_1 h_n r_n^3 + B_1 h_n r_n^2 + C_1 h_n r_n + D_1 h_n \\
+ A_2 h_n^2 r_n^3 + B_2 h_n^2 r_n^2 + C_2 h_n^2 r_n + D_2 h_n^2 \\
+ A_3 h_n^3 r_n^3 + B_3 h_n^3 r_n^2 + C_3 h_n^3 r_n + D_3 h_n^3 + \cdots .
\]

Let \(\delta = i \pmod{4}\) if \(\delta = 0\), then

\[
g_i(n) = \begin{cases} 
    a_i h_n^{[i/4]}, & \text{if } \delta = 0, \\
    a_i h_n^{[i/4+1]}.r_n^{4-\delta}, & \text{if } \delta \neq 0
\end{cases}
\]

\([i/4]\) represents the integral part of \(i/4\).

Thus, if \(\delta = 0\),

\[
\frac{g_{i+1}(n)}{g_i(n)} = c_i h_n^3 r_n^2 \to 0,
\]

if \(\delta \neq 0\),

\[
\frac{g_{i+1}(n)}{g_i(n)} - c_i \frac{1}{r_n} \to 0.
\]

We cannot assure convergence because for instance \(b_1 = b_2 = \frac{1}{2}\). Nevertheless we can drop hypotheses (H2) and (H3) because there are other sufficient conditions for the convergence acceleration (column by column) of the E-array based on the positivity of a certain determinant using the \(g_i\)'s, as stated in the next theorem of Matos and Prévost.

**Theorem 8** (Matos and Prévost [21, p. 398]). If \(S_n = S + a_1 g_1(n) + a_2 g_2(n) + \cdots \forall n\),

\[
g_i(n) \to 0 \quad \text{and} \quad \frac{g_{i+1}(n)}{g_i(n)} \to 0.
\]

If \(\forall i, \forall p, \forall n \geq N (g_0(n) \equiv 1)\)

\[
\begin{vmatrix}
g_i(p) & \cdots & g_i(n) \\
\vdots & \ddots & \vdots \\
g_{i+p}(n+p) & \cdots & g_i(n+p)
\end{vmatrix} \geq 0
\]

then \((E_k^{(n)} - S)/(E_k^{(n)} - S) \to 0, k = 0, 1, \ldots\).

**Remark.** In the cases we are studying \(g_i(n + 1)/g_i(n) \to h_i = 1/(2^n)\) and \(g_k(n)/g_k(n) = h_n^{(a_{k+1} - a_k)} r_n^{(b_{k+1} - b_k)}\).

So if, for instance (see Example 10 in Section 3.3),

\[
S_n = S + a_1 h_n \ln(h_n) + a_2 h_n \\
+ a_3 h_n^2 \ln(h_n) + a_4 h_n^2 \\
+ a_5 h_n^3 \ln(h_n) + a_6 h_n^3 + \cdots
\]
then

\[
g_{k+1}(n) = \begin{cases} 
\frac{1}{\ln(h_n)} \to 0, & k \text{ odd}, \\
\ln(h_n) \to 0, & k \text{ even}.
\end{cases}
\] 

(3.15)

In this case there are consecutive \( g_i \)'s with the same exponent \( c_i \) and therefore theorems 6 and 7 are not applicable. Nevertheless using the positivity of certain wronskians of the \( g_i \)'s and by the same type of arguments given by [21] it can be verified (see proof in [11]) that the conditions of Theorem 8 hold which means that even for this special case each column of the E-array converges faster than the previous one. Thus Theorem 8 has special relevance for the application of the E-algorithm to our problems. Details about the applicability of this theorem to asymptotic expansions where\(^1\) \( g_i(n) = (h_n)^{c_i}(-\ln h_n)^{\beta_i} \) are given in [11].

3.3. Numerical examples

The following numerical examples were suggested by the results obtained by [16] concerning the computation of the integral \( \Phi \) given by (2.52) which has been carried out with \( f(x) = x^{2m-1}\ln x \), \( m = 0, 1, 2, \ldots \) using Richardson extrapolation. In particular for \( m = 0 \), that is for \( f(x) = \ln(x)/x \), this extrapolation process is theoretically not applicable and gives poor results for \( m = 1 \). In the present work the above example is considered for the exponent \( m = 0 \) and for any real exponent. The numerical results given in [16] are therefore considerably improved by the use of the E-algorithm instead of Richardson extrapolation.

All the next examples suggest strict monotony in the terms of the initial sequence \( (S_n) \) obtained by application of the finite-difference schemes (2.12)-(2.13). This phenomenon is theoretically justified in the case of extrapolation algorithms for ordinary initial-value problems [12] and perhaps analogous extensions could be done to our more general problems.

We indicate by \( S \) what seems to be the best computed values obtained by the E-algorithm and \( S \) is given with all the apparently significant digits. References to the performances of the other algorithms used by the EXTRAP01 program are also given. For the \( \rho \)-algorithm, first and second generalizations of the \( \varepsilon \)-algorithm, the auxiliary sequence [5] used was \( x_n = \ln(n + 1) \), \( n = 0, 1, \ldots \) and for the Richardson extrapolation process \( x_n = 1/2^{n+1} \), \( n = 0, 1, \ldots \).

The results of all the examples have been obtained running the EXTRAP01 program over an initial sequence of 12 terms \( (S_n), n = 0, 1, \ldots, 11 \) and using 12 exponents \( c_i, \beta_i, i = 1, 2, \ldots, 12 \) for the respective error expansion extracted from Tables 1 and 2. In terms of the boundary-value problem (1.2)-(1.3) we have considered throughout the next examples

\[ p(x) = 1 + |x - 0.5| \] 

(3.16a)

and

\(^1\) The minus sign enables the exponentiation with positive base since we can consider without loss of generality \( h \in (0, 1) \).
\[
q(x) = \begin{cases} 
0, & \text{if } x < 0.5, \\
0.5, & \text{if } x = 0.5, \\
1, & \text{if } x > 0.5.
\end{cases} 
\] (3.16b)

The computations have been carried out in double precision. We give a table of results for each example consisting of the terms of \((S_n)\), the first column \(E_1(n)\), and the best column of the \(E\)-array we could compute. As generally we do not know the exact solution for our finite-difference schemes, such number of exact digits of each computed value \(E_k(n)\) can only be deduced by a careful inspection of the numerical results in the outputed arrays produced by the program. In this sense, the best column above mentioned is that one which shows consecutive entries with the maximum number of equal digits.

Example 9. Computation of \(\Psi = \int_0^1 f(x) u(x) \, dx\) with \(f(x) = x^{-1.4}\).

\[
\Psi_n = \Phi + B_1 h_n^{0.2} + C_1 h_n^{0.6} + B_2 h_n^{1.2} + C_2 h_n^{1.6} + A_2 h_n^2 \\
+ B_3 h_n^{2.2} + C_3 h_n^{2.6} + B_4 h_n^{3.2} + C_4 h_n^{3.6} + A_4 h_n^4 \\
+ B_5 h_n^{4.2} + C_5 h_n^{4.6} + B_6 h_n^{5.2} + C_6 h_n^{5.6} + A_6 h_n^6 + \cdots \quad n = 0, 1, \ldots, 11.
\]

In this case the convergence and convergence acceleration follow from Theorems 6 and 7.

For this problem, \(S = -9.3053023\). Curiously the starting values of \((S_n)\) do not have any exact digit and nothing indicates convergence at all. The \(E\)-algorithm is for this example the best of all competing algorithms. The nearest competitor is the \(\varepsilon\)-algorithm with approximately 6 significant digits. Richardson extrapolation and the \(\rho\)-algorithm produce answers with approximately 1 significant digit, while the curvature-type algorithms and the first and second generalizations of the \(\varepsilon\)-algorithm, with approximately 0 significant digits, are the worst (see Table 3).

<table>
<thead>
<tr>
<th>(n)</th>
<th>(S_n)</th>
<th>(E(n, 1))</th>
<th>(E(n, 8))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.3316382988658410E + 00</td>
<td>-0.3985228897949082E + 01</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-0.8045935504341072E + 00</td>
<td>-0.5465459805083570E + 01</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-0.1407940070032363E + 01</td>
<td>-0.6621520734566416E + 01</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-0.2082835159642257E + 01</td>
<td>-0.7467568306622714E + 01</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.2779885842003097E + 01</td>
<td>-0.8063198035703753E + 01</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.3463807638905090E + 01</td>
<td>-0.8472776426766524E + 01</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-0.4112215835972678E + 01</td>
<td>-0.8750316386124030E + 01</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-0.4712615347906546E + 01</td>
<td>-0.8936639032795661E + 01</td>
<td>-0.930529600909725E + 01</td>
</tr>
<tr>
<td>8</td>
<td>-0.5294128421354646E + 01</td>
<td>-0.9060975985996890E + 01</td>
<td>-0.9305302389158657E + 01</td>
</tr>
<tr>
<td>9</td>
<td>-0.5751523056590090E + 01</td>
<td>-0.9143625720649000E + 01</td>
<td>-0.9305302363584236E + 01</td>
</tr>
<tr>
<td>10</td>
<td>-0.6190628841749614E + 01</td>
<td>-0.9198424917029229E + 01</td>
<td>-0.9305302300008506E + 01</td>
</tr>
<tr>
<td>11</td>
<td>-0.6579986354823701E + 01</td>
<td>-0.9305302300008506E + 01</td>
<td></td>
</tr>
</tbody>
</table>
Table 4

<table>
<thead>
<tr>
<th>n</th>
<th>$S_n$</th>
<th>$E(n, 1)$</th>
<th>$E(n, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.1320280343923705E + 00</td>
<td>-0.9499302892909710E + 05</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-0.1948218012718484E + 00</td>
<td>-0.3822740574993539E + 00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-0.2416849582633046E + 00</td>
<td>-0.3344176041461751E + 00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-0.27259810907005E + 00</td>
<td>-0.3233182345107727E + 00</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.2916167845790552E + 00</td>
<td>-0.3175075206999569E + 00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.3028611754474218E + 00</td>
<td>-0.3184082889437546E + 00</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-0.3093391458159326E + 00</td>
<td>-0.3178894997065765E + 00</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-0.316762258346867E + 00</td>
<td>-0.3176765637294646E + 00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-0.3150480157348864E + 00</td>
<td>-0.3175864863619101E + 00</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>-0.316762258346867E + 00</td>
<td>-0.3175475143493360E + 00</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-0.3171283295097570E + 00</td>
<td>-0.3175303569389985E + 00</td>
<td></td>
</tr>
</tbody>
</table>

Example 10. Computation of $u(0.5)$ with $f(x) = \ln(x)/x$.

$$u_h^k = u + A_1 h_n \ln h_n + B_1 h_n$$

$$+ A_2 h_n^2 \ln h_n + B_2 h_n^2$$

$$+ A_3 h_n^3 \ln h_n + B_3 h_n^3 + \cdots \quad n = 0, 1, \ldots, 11.$$  

For this problem, $S = 0.317516315$. With approximately 10 significant digits it seems that the $\varepsilon$-algorithm gives numerical results a little more accurate than the $E$-algorithm. The $\rho$-algorithm, the first and second generalizations of the $\varepsilon$-algorithm, and the Richardson extrapolation with approximately 3 or 4 significant digits give the worst numerical results (see Table 4).

Example 11. Computation of $u(0.5)$ with $f(x) = 1/\sin x$.

$$u_h^k = u + A_1 h_n + A_2 h_n^2 + A_3 h_n^3 + \cdots, \quad n = 0, 1, \ldots, 11.$$  

In this case the convergence and convergence acceleration follow from Theorems 4 and 5.

For this problem, $S = -0.253282155845$. All the algorithms except the $\rho$ and first and second generalizations of the $\varepsilon$-algorithm give results with more than 10 significant digits. Here it seems that the best is the iterated $\Delta^2$-Aitken with approximately 13 significant digits (see Table 5).
Table 5

<table>
<thead>
<tr>
<th>n</th>
<th>S_n</th>
<th>E(n, 1)</th>
<th>E(n, 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.1986504421841417E+00</td>
<td>-0.2424291464613039E+00</td>
<td>-0.2501503155917028E+00</td>
</tr>
<tr>
<td>1</td>
<td>-0.2205397943227228E+00</td>
<td>-0.2524713687578090E+00</td>
<td>-0.2532840460675486E+00</td>
</tr>
<tr>
<td>2</td>
<td>-0.2353450549572128E+00</td>
<td>-0.2524713687578090E+00</td>
<td>-0.2532840460675486E+00</td>
</tr>
<tr>
<td>3</td>
<td>-0.2439082118575109E+00</td>
<td>-0.253078674863685E+00</td>
<td>-0.2532822074560864E+00</td>
</tr>
<tr>
<td>4</td>
<td>-0.2484934434219397E+00</td>
<td>-0.2532313877655993E+00</td>
<td>-0.253281568156024E+00</td>
</tr>
<tr>
<td>5</td>
<td>-0.2508624155937695E+00</td>
<td>-0.2532694896983369E+00</td>
<td>-0.253281558605995E+00</td>
</tr>
<tr>
<td>6</td>
<td>-0.252069526460532E+00</td>
<td>-0.2532789932712224E+00</td>
<td>-0.253281558454710E+00</td>
</tr>
<tr>
<td>7</td>
<td>-0.2526274730086378E+00</td>
<td>-0.2532813657865026E+00</td>
<td>-0.253281558454710E+00</td>
</tr>
<tr>
<td>8</td>
<td>-0.2529769193975702E+00</td>
<td>-0.2532819584038060E+00</td>
<td>-0.253281558454710E+00</td>
</tr>
<tr>
<td>9</td>
<td>-0.2531294389006881E+00</td>
<td>-0.2532821064949310E+00</td>
<td>-0.25328155845387E+00</td>
</tr>
<tr>
<td>10</td>
<td>-0.2532057726978096E+00</td>
<td>-0.2532821435088489E+00</td>
<td>-0.25328155845387E+00</td>
</tr>
<tr>
<td>11</td>
<td>-0.2532439581033292E+00</td>
<td>-0.2532822074560864E+00</td>
<td>-0.25328155845387E+00</td>
</tr>
</tbody>
</table>

Example 12. Computation of $\Phi = \int_{0}^{1} f(x) u(x) \, dx$ with $f(x) = \ln(x) / x$.

$$
\Phi^h = \Phi + A_1 h_n (\ln(h_n))^3 + B_1 h_n (\ln(h_n))^2 + C_1 h_n \ln(h_n) + D_1 h_n
+ A_2 h_n^2 (\ln(h_n))^3 + B_2 h_n^2 (\ln(h_n))^2 + C_2 h_n^2 \ln(h_n) + D_2 h_n^2
+ A_3 h_n^3 (\ln(h_n))^3 + B_3 h_n^3 (\ln(h_n))^2 + C_3 h_n^3 \ln(h_n) + D_3 h_n^3 + \cdots,
$$

For this problem, $S = -3.367885$. Now the only algorithm comparable with the E-algorithm is the iterated $\Delta^2$-Aitken process giving approximately 7 significant digits. Here the curvature-type algorithms and the Richardson extrapolation produce numerical answers with approximately 3 significant digits, while the $p$, first and second generalizations of the $e$-algorithm are even worst extrapolation methods (see Table 6).

Table 6

<table>
<thead>
<tr>
<th>n</th>
<th>S_n</th>
<th>E(n, 1)</th>
<th>E(n, 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.9151485979394312E-01</td>
<td>-0.5617651905993433E-02</td>
<td>-0.3369432685935006E+01</td>
</tr>
<tr>
<td>1</td>
<td>-0.3829123948937527E+00</td>
<td>-0.2785502265492552E+00</td>
<td>-0.3368593496526689E+01</td>
</tr>
<tr>
<td>2</td>
<td>-0.8376679471358205E+00</td>
<td>-0.1973650064844888E+01</td>
<td>-0.3368677297605211E+01</td>
</tr>
<tr>
<td>3</td>
<td>-0.1358282393798915E+01</td>
<td>-0.226899664733236E+02</td>
<td>-0.3368677297605211E+01</td>
</tr>
<tr>
<td>4</td>
<td>-0.1856814759116104E+01</td>
<td>-0.4981548367748933E+01</td>
<td>-0.3369432685935006E+01</td>
</tr>
<tr>
<td>5</td>
<td>-0.2281778529890169E+01</td>
<td>-0.3902401522810015E+01</td>
<td>-0.33689740139497941E+01</td>
</tr>
<tr>
<td>6</td>
<td>-0.2615656877968563E+01</td>
<td>-0.3589470139497941E+01</td>
<td>-0.3368677297605211E+01</td>
</tr>
<tr>
<td>7</td>
<td>-0.2862659075499338E+01</td>
<td>-0.3468934853110055E+01</td>
<td>-0.3367769577082375E+01</td>
</tr>
<tr>
<td>8</td>
<td>-0.3037163034869613E+01</td>
<td>-0.3415617327994716E+01</td>
<td>-0.336766981099757E+01</td>
</tr>
<tr>
<td>9</td>
<td>-0.315604648229899E+01</td>
<td>-0.339115922304679E+01</td>
<td>-0.3367880005000814E+01</td>
</tr>
<tr>
<td>10</td>
<td>-0.323469193784913E+01</td>
<td>-0.3379419919988079E+01</td>
<td>-0.336783437954634E+01</td>
</tr>
</tbody>
</table>
Remark. In the case of Examples 10 and 12 the convergence of the F-algorithm should be proved on the base of Theorem 8. With this purpose we must verify that the determinantal condition (3.14) is satisfied. Such study is made by Graça in [11].

Acknowledgements

The authors wish to express their gratitude to Professor Claude Brezinski for helpful suggestions made during the preparation of this paper.

Appendix. Fortran code for $g_i(k - 1)$ and $g_k - 1(i - 1)$ needed by EALGO subroutine

```fortran
C ... COMPUTATION OF $G_f(K - 1)$
M = MAX0(I, K-2)
IF(K.GT.MAXCOL + 1) M = MAXCOL
DO 10 I = 1, M
DN0 = DBLE(N0)
AHKL1 = 1.D0/(DN0*2.DO**(K - 1))
ARKL1 = DLOG(AHKL1)
C ... NP1 IS A VECTOR FOR THE H'S AND NP2 IS A VECTOR
C FOR THE LOG(H)'S
ALPHA1 = NP1(1)
BETA1 = NP2(1)
IF(ARKL1 .LT. 0.DO) THEN
LNHBETAI = -(DABS(ARKL1)**BETA1)
ELSE
LNHBETAI = ARKL1**BETA1
ENDIF
GINIT(I) = AHKL1**ALPHA1*LNHBETAI
10 CONTINUE

C ... COMPUTATION OF $G_{K-1}(I - 1)$
IF(K .GE. 3 .AND. K .LE. MAXCOL + 1) THEN
DO 20 I = 1, K
IL1 = I - 1
MMI = M + I
DN0 = DBLE(N0)
AHIL1 = 1.D0/(DN0*2.DO**(IL1))
ARIL1 = DLOG(AHIL1)
ALPHA1 = NP1(K - 1)
BETA1 = NP2(K - 1)
IF(ARIL1 .LT. 0.DO) THEN
LNHBETAI = -(DABS(ARIL1)**BETA1)
```

ELSE
LNHBETAI = ARIL1**BETAI
ENDIF
GINIT(1) = AHIL1**ALPHAI*LNHBETAI
20 CONTINUE

References