EXTRAPOLATION METHODS FOR THE NUMERICAL SOLUTION OF NONLINEAR FREDHOLM INTEGRAL EQUATIONS

CLAUDE BREZINSKI AND MICHELA REDIVO–ZAGLIA

ABSTRACT. In this paper, we want to exemplify the use of extrapolation methods (namely Shanks transformations, the recursive algorithms for their implementation, and the freely available corresponding MATLAB software) in the solution of nonlinear Fredholm integral equations of the second kind. Extrapolations methods are well–known in some domains of numerical analysis and applied mathematics, but, unfortunately, they are not frequently used in other domains. Thus, after presenting the most simple iterative method for the solution of Fredholm equations, we will show how the sequence it produces can be accelerated (under some assumptions) and also how the underlying system of nonlinear equations generated by it can be solved quite efficiently by a restarting method. Numerical examples and comparisons with other methods demonstrate the usefulness of these procedures.

1. Introduction and motivation. Extrapolation methods (also named convergence acceleration methods) form a particular chapter of numerical analysis [17, 70]. They have been successfully used in several branches of applied mathematics. However, they do not seem to be well known from researchers working on integral equations. The aim of this paper is to present one of these extrapolation methods which is particularly well adapted to their solution, and can render many services in this domain. We will focus on Fredholm integral equations which arise, for example, in signal processing, linear modeling, inverse problems, diffraction problems, scattering, conformal mapping, water wave; see [67].

1991 AMS *Mathematics subject classification.* 65R20, 65B05, 65B99, 65H10.

Keywords and phrases. Integral equations, acceleration techniques; sequence transformations, Shanks transformations, solution of equations.

The work of C.B. was supported by the Labex CEMPI (ANR-11-LABX-0007-01.

The work of M.R.-Z. was partially supported by the University of Padua, Project 2014 No. CPDA143275.

Received by the editors November 13, 2017.

When a sequence (of numbers, vectors, matrices, tensors or, more generally elements of a vector space) is slowly converging it can be transformed, by a *sequence transformation*, into a new sequence which, under some assumptions, converges faster to the same limit. Since such transformations are based on the idea of extrapolation, they are also often named *extrapolation methods*. The most well-known such methods are, in the scalar case, Romberg's method for accelerating the trapezoidal rule for the evaluation of a definite integral, and Aitken's Δ^2 process. Based to Aitken's process, Steffensen's method for solving a nonlinear equation in one unknown is also well established. Some of these methods have been extended to non–scalar sequences and they have been successfully used in the solution of a number of problems in numerical analysis and applied mathematics [2, 14, 17, 18, 20, 25, 28, 29, 35–38, 40, 47, 50, 55, 59, 61, 62, 66, 69].

In this paper, we propose a new approach for solving Fredholm equations by using a fixed point acceleration strategy. This kind of strategy is nowadays well known in the literature (and for various fields of application) but seems never have been applied to the field of Fredholm equations.

Our method simply consists in evaluating the integral by a quadrature rule, then solving iteratively the corresponding system of nonlinear equations obtained by collocation, and finally in accelerating these iterations by a sequence transformation implemented via a recursive algorithm. We also use a restarting procedure for this nonlinear system. Thus, our procedures are quite simple and easy to implement while many of the techniques encountered elsewhere are more sophisticated and difficult to implement.

It is not our purpose to produce a method able to compete successfully with all those which can be found in the literature. Moreover, it is clear that, since the quadrature rule we used for approximating the integral is simply the trapezoidal rule, more precise quadrature rules will probably lead to better results. We only want to introduce a new efficient method in the toolbox of researchers working on the numerical solution of linear and nonlinear Fredholm integral equations of the second kind, and to exemplify the use of extrapolation methods (namely Shanks transformations and the corresponding recursive algorithms for their implementation) in this solution. We show that Shanks transformations are quite useful in accelerating the convergence of Picard iterations (or, more generally, relaxation methods) and that, coupled with a restarting procedure, they are an interesting way for solving the underlying system of nonlinear equations. The interest of an extrapolation method is that it can be used for accelerating the convergence of a sequence produced by any iterative method used for solving Fredholm equations and, by a restarting procedure, it leads to a simple method for solving fixed point problems. Thus, these methods can also be used for the acceleration of projection methods (Galerkin and collocation) [4]. Among these methods are the Minimal Polynomial Extrapolation (MPE) [23], the Modified Minimal Extrapolation (MMPE) [11, 54], the Reduced Rank Extrapolation (RRE) [27, 48], the Vector Epsilon Algorithm (VEA) [72], the Topological Epsilon Algorithm (TEA) [11], and other extrapolation methods. Moreover, among previously cited extrapolation methods, only the methods given in this paper can be easily recursively implemented. All others, except the VEA and the MMPE, require the solution of a system of linear equations at each step. They have been tested on many numerical examples, and the results they produce are quite comparable. Let us mention that various generalizations of Padé approximants, some of them being connected to the ε–algorithm(s), were also used in the solution of integral equations [31, 32, 64].

Automatic programs for Fredholm integral equations can be found in [3, 6], but only for the linear case. Thus, the interest of this work also lies in the MATLAB software which is freely available and easy to use while this is not the case for other methods presented in the literature. The MATLAB files *Extrapolation for nonlinear Fredholm integral equations*, producing our numerical results, can be downloaded from the Matlab File Exchange site

`www.mathworks.com/matlabcentral/fileexchange/`.

They also allow to try them with other values of the various parameters, and make new experiments. Moreover, additional integral equations have been included into the package. For the sequence transformations needed, these programs use the MATLAB toolbox `EPSfun` which is freely available as `na44` from the `numeralgo` library of `netlib` [21].

Section 2 begins by fixing our notations for the Fredholm integral equation to be solved. In Section 2.1, we describe the approximation scheme that will be used for solving it. It simply consists in approximating the integral by a quadrature formula, namely the trapezoidal rule. The system of nonlinear equations obtained by this approximation scheme is given in Section 2.2, and the Picard iterative method (or the relaxation method) for its solution is explained in Section 2.3. In Section 3, we present the sequence transformations that will be used for accelerating the sequence produced by the iterative method and for solving the system of nonlinear equations. We also discuss the algorithms for their implementation. Section 4 is devoted to numerical examples showing the effectiveness of these procedures and comparisons with other methods.

2. Fredholm integral equations. We consider the following nonlinear *Fredholm integral equation of the second kind* with a given kernel K

$$u(t) = \int_a^b K(t,x,u(x))\,dx + f(t), \quad t \in [a,b]. \tag{1}$$

This equation is also often said to be a *Urysohn integral equation*. It is assumed that K is neither singular nor weakly singular.

We assume that $f, u \in C[a,b]$ and $K \in C([a,b] \times [a,b] \times \mathbb{R})$. If K satisfies a uniform Lipschitz condition

$$\|K(t,x,v) - K(t,x,w)\| \leq L\|v - w\|, \ \forall t, x \in [a,b], \ \text{and } v, w \in C[a,b],$$

then this integral equation has a unique solution in $C[a,b]$ if $L(b-a) < 1$ [7, 74].

This equation will be solved by successive approximations (Picard iterations) starting from an initial approximation $u^{(0)}(t)$

$$u^{(n+1)}(t) = \int_a^b K(t,x,u^{(n)}(x))\,dx + f(t),$$

or, more generally, by the relaxation method

$$u^{(n+1)}(t) = u^{(n)}(t) - \alpha \left[u^{(n)}(t) - \int_a^b K(t,x,u^{(n)}(x))\,dx - f(t) \right], \tag{2}$$

where α is a parameter different from 1 to be adjusted for convergence. As we will see below, these iterations need not converge for applying them our extrapolation schemes. In particular, the classical case $\alpha = 1$ (Picard iterations) does not always lead to convergence. In some examples, this choice leads to divergence, while, in some others, it converges rapidly, then stagnates, and one cannot see the benefit brought by the ε-algorithm. However, the relaxation method (2) will do so if the right hand side is a contraction with a Lipschitz constant $L = |1-\alpha| + |\alpha| L(b-a) < 1$. It is not our purpose here to discuss the choice of α (see, for example, [12]), and it will be experimentally taken in our numerical examples.

The parameter α can also be modified at each iteration, and replaced by α_n. Such an iterative method is due to Mann [45], and convergence results occur under various assumptions. In particular, let T be a mapping from a nonempty, convex subset of a real Banach space into itself, then the Mann iterations $v_{n+1} = (1-\alpha_n)v_n + \alpha_n T(v_n)$ converge to a fixed point of $T : v \longmapsto T(v)$ if the sequence (α_n) converges to zero and if the series $\sum \alpha_n$ diverges (see, for example [30, p. 83] and [49]). Dynamic relaxation is another procedure for finding a good value for this parameter (see [40] or [2]) as well as Richardson acceleration [13, Chap. 7]. Another technique consists in accelerated refinement [26]. Many other fixed point methods, such as those described in [12,15,40,42,46,51,57], can be considered as dynamic relaxation procedures.

Similarly, *Fredhlom integral equations of the first kind* are written as

$$f(t) = \int_a^b K(t, x, u(x))\, dx$$

and they can be solved correspondingly by the iterations

$$u^{(n+1)}(t) = u^{(n)}(t) + \alpha \left[f(t) - \int_a^b K(t, x, u^{(n)}(x))\, dx \right].$$

The aim of this paper is to show that some extrapolation methods can be quite useful either in accelerating the convergence of the relaxation method (2) or, more directly, for solving the system of nonlinear equations obtained from (1) after discretization (see below).

2.1. The approximation. For solving a Fredholm integral equation, a standard way (see [52, pp. 18ff.]) is to approximate the integral contained in it by a quadrature formula

$$\int_a^b K(t,x,u(x))\,dx \simeq \sum_{j=0}^{p} w_j^{(p)} K(t, x_j^{(p)}, u(x_j^{(p)})), \tag{3}$$

where $x_0^{(p)}, \ldots, x_p^{(p)}$ are $p+1$ points in $[a,b]$, and where the upper index p denotes the dependence on the number of points chosen. Remember that the weights $w_j^{(p)}$ are strictly positive and sum up to $b-a$. Thus (1) is approximated by

$$u_p(t) = \sum_{j=0}^{p} w_j^{(p)} K(t, x_j^{(p)}, u_p(x_j^{(p)})) + f(t). \tag{4}$$

Such a method is called a *quadrature method* or a *Nyström method*.

2.2. The system of nonlinear equations. We will now approximate the solution u_p by collocation at the points $t_i^{(p)} = x_i^{(p)}$ for $i = 0, \ldots, p$. For a fixed value of p, we set for simplicity $t_i = x_i = t_i^{(p)} = x_i^{(p)}, f_i = f(t_i), w_i^{(p)} = w_i$, and we determine approximations u_i of $u_p(t_i)$, $i = 0, \ldots, p$, as the solution of the system of $p+1$ nonlinear equations

$$u_i = \sum_{j=0}^{p} w_j K(t_i, t_j, u_j) + f_i, \quad i = 0, \ldots, p. \tag{5}$$

The function

$$y(t) = \sum_{j=0}^{p} w_j K(t, t_j, u_j) + f(t)$$

interpolates the discrete solution $u_p(t)$ at the points $t_i, i = 0, \ldots, p$. This formula is known as the *Nyström interpolation formula*. Its error is governed by the one of the numerical integration method used for (3), see [4].

2.3. The iterative scheme. The solution of the system of nonlinear equations (5) can be obtained by Newton or quasi–Newton, or Broyden methods. However such methods are expensive to implement [4].

Thus, for solving the nonlinear system, we use the fixed point iterative scheme (Picard iterations), for $n = 0, 1, \ldots$ until convergence

$$u_i^{(n+1)} = \sum_{j=0}^{p} w_j K(t_i, t_j, u_j^{(n)}) + f_i, \quad i = 0, \ldots, p, \tag{6}$$

or, more generally, according to (2), the relaxation scheme

$$u_i^{(n+1)} = u_i^{(n)} - \alpha \left\{ u_i^{(n)} - \sum_{j=0}^{p} w_j K(t_i, t_j, u_j^{(n)}) - f_i \right\}, \quad i = 0, \ldots, p, \tag{7}$$

where α is a parameter to adjust for convergence, and $u_i^{(0)}, i = 0, \ldots, p$, is the initial approximation of the solution at the points t_i.

A similar iterative scheme can be used for Fredholm equations of the first kind.

Thus we have two nested approximation methods: the exact solution u, evaluated at the points t_i, that is $\mathbf{u} = (u(t_0), \ldots, u(t_p))^T$ is first approximated through (4) by $\mathbf{u}_p = (u_p(t_0), \ldots, u_p(t_p))^T$ for a fixed value of p, where $u_p(t_i)$ approximates $u(t_i)$, which is itself approximated by the iterates $\mathbf{u}^{(n)} = (u_0^{(n)}, \ldots, u_p^{(n)})^T$, where $u_i^{(n)}$ is the approximation of $u_p(t_i)$ obtained at the nth iteration of the iterative method (6) or (7) for the solution of the nonlinear system.

It holds, for $i = 0, \ldots, p$,

$$\begin{aligned} u_i^{(n+1)} - u_p(t_i) &= (1-\alpha)(u_i^{(n)} - u_p(t_i)) \\ &\quad + \alpha \sum_{j=0}^{p} w_j [K(t_i, t_j, u_j^{(n)}) - K(t_i, t_j, u_p(t_j))]. \end{aligned}$$

The iterative method described above is quite elementary and cheap and it may not converge, or its convergence could be quite slow. However, the sequence of vectors $(\mathbf{u}^{(n)})$ can be directly accelerated by a suitable method or the nonlinear system can be solved by the Generalized Steffensen Method (a quasi–Newton method based on an acceleration procedure) [9, 10, 21].

3. Shanks transformations and the ε–algorithms. In this Section, we will not come back to the history of Shanks transformation [58] for scalar sequences and to the ε–algorithm for its recursive implementation [71]. We will only discuss the case of sequences of elements of a vector space E. Of course, there exist many methods for accelerating the convergence of scalar sequences, and it is often quite difficult to know, a priori, which one will give the best results. However, as stated in [33], the *ε–algorithm* (of Wynn [71]) *is arguably the best all–purpose method for accelerating the convergence.* This remark is based on the fact that this algorithm gives the exact limit of sequences which behave as a combination of exponentials (a case frequently encountered in practice), and on the numerical experiments performed for many years with various acceleration methods. The scalar ε–algorithm was extended to vector sequences by Wynn [72] and, later, to sequences in a general vector space by Brezinski [11] as will be explained below. Reviews on these methods can be found in [17, 60, 69, 70] and, more recently, in [22].

Let $(\mathbf{S}_n)$ be a sequence of elements of a vector space E on $\mathbb{R}$ or $\mathbb{C}$. We assume that, for all n, this sequence satisfies the homogeneous linear difference equation of order k

$$a_0(\mathbf{S}_n - \mathbf{S}) + \cdots + a_k(\mathbf{S}_{n+k} - \mathbf{S}) = \mathbf{0} \in E, \tag{8}$$

where $\mathbf{S} \in E$ and the a_i's are scalar coefficients with $a_0 a_k \neq 0$, $a_0 + \cdots + a_k \neq 0$ (otherwise $\mathbf{S}$ is not uniquely determined by this relation). If the sequence $(\mathbf{S}_n)$ converges, $\mathbf{S}$ is its *limit*. Otherwise, it is called its *antimilit*. It does not restrict the generality to assume that the a_i's sum up to 1. This is the *normalization condition* that we will consider.

Obviously, if the coefficients a_i are known, $\mathbf{S} = a_0\mathbf{S}_n + \cdots + a_k\mathbf{S}_{n+k}$ for all n. If they are unknown, they have to be computed. For that purpose, we transform the equation (8) in E into k scalar relations. Let $\mathbf{y}$ be an element of E^*, the algebraic dual space of E, that is the space of linear functionals on E. We denote by $\langle \cdot, \cdot \rangle$ the duality product between E^* and E.

Writing (8) for the indices $n+1$ and n, subtracting, and applying $\mathbf{y}$, we obtain, for all n,

$$a_0\langle \mathbf{y}, \Delta\mathbf{S}_n\rangle + \cdots + a_k\langle \mathbf{y}, \Delta\mathbf{S}_{n+k}\rangle = 0 \in \mathbb{C} \text{ or } \mathbb{R},$$

where Δ is the usual forward difference operator defined by $\Delta\mathbf{S}_n = \mathbf{S}_{n+1} - \mathbf{S}_n$.

Solving the system of linear equations

$$\left.\begin{array}{rcccl} a_0 & + \cdots + & a_k & = 1 & \\ a_0\langle \mathbf{y}, \Delta\mathbf{S}_{n+i}\rangle & + \cdots + & a_k\langle \mathbf{y}, \Delta\mathbf{S}_{n+k+i}\rangle & = 0, & i = 0, \ldots, k-1 \end{array}\right\} \tag{9}$$

gives the coefficients a_i and thus we can compute $\mathbf{S}$.

If the sequence $(\mathbf{S}_n)$ does not satisfy the relation (8), we still assume that it holds (this kind of assumption is the basis for constructing any convergence acceleration or extrapolation method [17]). Then, the preceding system for the a_i's can still be solved but its solution now depends on k and n, and the linear combination giving $\mathbf{S}$ can still be computed. Thus, we set

$$\widehat{e}_k(\mathbf{S}_n) = a_0\mathbf{S}_n + \cdots + a_k\mathbf{S}_{n+k} \tag{10}$$

which, for a fixed value of k, defines the sequence transformation $(\mathbf{S}_n) \longmapsto (\widehat{e}_k(\mathbf{S}_n))$. This transformation is called the *topological Shanks transformation* (since, for speaking about convergence, E has to be a topological vector space). By construction, if $(\mathbf{S}_n)$ satisfies (8), then, $\forall n, \widehat{e}_k(\mathbf{S}_n) = \mathbf{S}$. This set of sequences is called the *kernel* of the topological Shanks transformation. It includes sequences which behave as a sum of exponential functions, a common feature to many iterative procedures, which explains its efficiency in a number of cases [16].

From (9) and (10), it holds

$$\widehat{e}_k(\mathbf{S}_n) = \frac{\begin{vmatrix} \mathbf{S}_n & \cdots & \mathbf{S}_{n+k} \\ \langle \mathbf{y}, \Delta\mathbf{S}_n\rangle & \cdots & \langle \mathbf{y}, \Delta\mathbf{S}_{n+k}\rangle \\ \vdots & & \vdots \\ \langle \mathbf{y}, \Delta\mathbf{S}_{n+k-1}\rangle & \cdots & \langle \mathbf{y}, \Delta\mathbf{S}_{n+2k-1}\rangle \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ \langle \mathbf{y}, \Delta\mathbf{S}_n\rangle & \cdots & \langle \mathbf{y}, \Delta\mathbf{S}_{n+k}\rangle \\ \vdots & & \vdots \\ \langle \mathbf{y}, \Delta\mathbf{S}_{n+k-1}\rangle & \cdots & \langle \mathbf{y}, \Delta\mathbf{S}_{n+2k-1}\rangle \end{vmatrix}}. \tag{11}$$

This is the *first topological Shanks transformation.* The determinant in the numerator denotes the linear combination of $\mathbf{S}_n, \ldots, \mathbf{S}_{n+k}$ obtained by developing it with respect to its first row by the classical rule for expanding a determinant. Replacing this first row by $\mathbf{S}_{n+k}, \ldots, \mathbf{S}_{n+2k}$

leads to the *second topological Shanks transformation* defined by

$$\widetilde{e}_k(\mathbf{S}_n) = a_0\mathbf{S}_{n+k} + \cdots + a_k\mathbf{S}_{n+2k},$$

where the coefficients a_i are the same as for the first transformation. This could lead to a better result since the linear combination uses elements of the sequence $(\mathbf{S}_n)$ with higher indices which are usually closer to the limit $\mathbf{S}$. As we will see below, its recursive implementation is also easier and cheaper.

These transformations were introduced in [11]. They can be implemented by recursive algorithms (the *topological ε–algorithms*, TEAs in short). The rules of these algorithms were recently greatly simplified, thus leading to the *simplified topological ε–algorithms*, STEAs in short [19]. There is now only one rule instead of two, the functional $\mathbf{y}$ has no longer to be used in the rule of the algorithm but only in its initialization, the necessary storage has been much reduced, and the numerical stability can be partly controlled. These new algorithms also allow to prove some theoretical results on the convergence and the acceleration of the transformation [19]. The corresponding software and applications to the acceleration of vector and matrix sequences, the solution of systems of nonlinear vector and matrix equations, and the computation of matrix functions were given in [21]. It is these algorithms and this software contained in `EPSfun` that we used to produce the numerical examples of Section 4. Let us now present them.

We first have to begin by the scalar ε–algorithm of Wynn [71]. We consider the real or complex sequence (s_n) (notice the lowercase letters instead of the capital ones). This algorithm consists in the rule

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

with $\varepsilon_{-1}^{(n)} = 0$ and $\varepsilon_0^{(n)} = s_n$ for $n = 0, 1, \ldots$ This algorithm implements the scalar Shanks transformation of the sequence (s_n) which is given by the same ratio of determinants as in (11) (but now denoted by $e_k(s_n)$) where s_i replaces $\mathbf{S}_i$ and Δs_i replaces $\langle \mathbf{y}, \Delta\mathbf{S}_i\rangle$, and we obtain $\varepsilon_{2k}^{(n)} = e_k(s_n)$.

These $\varepsilon_k^{(n)}$'s are displayed in a two dimensional array called the *ε–array*, and the rule of the algorithm relates the quantities located at the summits of a rhombus as showed in Table 1.

$$
\begin{array}{lllll}
\varepsilon_{-1}^{(0)}=0 & & & & \\
 & \varepsilon_{0}^{(0)}=s_0 & & & \\
\varepsilon_{-1}^{(1)}=0 & & \varepsilon_{1}^{(0)} & & \\
 & \varepsilon_{0}^{(1)}=s_1 & & \varepsilon_{2}^{(0)} & \\
\varepsilon_{-1}^{(2)}=0 & & \varepsilon_{1}^{(1)} & & \ddots \\
\vdots & \varepsilon_{0}^{(2)}=s_2 & \vdots & & \\
\vdots & \vdots & \vdots & & \varepsilon_{2k}^{(0)} \\
\vdots & \vdots & \vdots & & \\
\vdots & \varepsilon_{0}^{(2k-1)}=s_{2k-1} & \vdots & \iddots & \\
\varepsilon_{-1}^{(2k)}=0 & & \varepsilon_{1}^{(2k-1)} & & \\
 & \varepsilon_{0}^{(2k)}=s_{2k} & & &
\end{array}
\qquad
\begin{array}{ccccc}
 & \nearrow & \varepsilon_{k}^{(n)} & \searrow & \\
\varepsilon_{k-1}^{(n+1)} & & & & \boxed{\varepsilon_{k+1}^{(n)}} \\
 & \searrow & \varepsilon_{k}^{(n+1)} & \nearrow &
\end{array}
$$

TABLE 1. The ε-array (left) and the rhombus rule (right).

For a vector sequence $(\mathbf{S}_n)$, the first topological Shanks transformation can be implemented by the following recursive algorithm, called the *first simplified topological ε–algorithm* (STEA1)

$$
\begin{aligned}
\widehat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \widehat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}} (\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} - \widehat{\boldsymbol{\varepsilon}}_{2k}^{(n)}), \quad k,n = 0,1,\ldots, \\
\widehat{\boldsymbol{\varepsilon}}_{0}^{(n)} &= \mathbf{S}_n, \quad n = 0,1,\ldots,
\end{aligned}
$$

where the $\varepsilon_k^{(n)}$'s are obtained by applying Wynn's scalar ε–algorithm to the sequence $(s_n = \langle \mathbf{y}, \mathbf{S}_n \rangle)$, and we obtain $\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n)} = \widehat{e}_k(\mathbf{S}_n)$ as given by (11).

The second topological Shanks transformation is implemented by the *second simplified topological ε–algorithm* (STEA2) with the same initializations as for the first one

$$\widetilde{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} = \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}} (\widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}),$$

where the $\varepsilon_k^{(n)}$'s are the same as for the STEA1, and we get $\widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n)} = \widetilde{e}_k(\mathbf{S}_n)$ but for the second topological Shanks transformation, that is after replacing the first row of the numerator of (11) by $\mathbf{S}_{n+k}, \ldots, \mathbf{S}_{n+2k}$.

The initializations of both algorithms are $\widehat{\boldsymbol{\varepsilon}}_0^{(n)} = \widetilde{\boldsymbol{\varepsilon}}_0^{(n)} = \mathbf{S}_n$, $n = 0, 1, \ldots$ Let us mention that, thanks to the strategy used for its implementation (which was originally described by Wynn for his scalar ε–algorithm [73]) the STEA2 is much cheaper in term of storage than the STEA1. Moreover other equivalent formulæ exist for both the algorithms. Numerical test showed that the results are almost equivalent and, thus, here we indicate only one of them. For details see [19, 21].

There are two different ways of using these algorithms. The simplest one is to use the STEA1 or the STEA2 for accelerating a sequence $(\mathbf{S}_n)$ of vectors or matrices or tensors or, more generally, elements of a vector space E. It is named the *Acceleration Method* and denoted AM below. Usually, the user fixes the maximal even column $2k$ to reach in the ε-array. For reaching the first term of the column $2k$, $2k+1$ terms are needed, that is $2k$ evaluations of the system. Then, one more evaluation leads to the next term of this column. In this way, the algorithm furnishes the staircase along the main descending diagonal until the column $2k$ has been reached. This is showed in Table 2 where $\boldsymbol{\varepsilon}$ stands for $\widehat{\boldsymbol{\varepsilon}}$ or $\widetilde{\boldsymbol{\varepsilon}}$.

The second way of using the algorithms is a restarted version of them for the solution of a system of linear or nonlinear equations $\mathbf{S} = F(\mathbf{S})$, where $F : \mathbb{R}^m \longmapsto \mathbb{R}^m$. Starting from a given $\mathbf{x}_0$, we fix $2k$, we set $\mathbf{S}_0 = \mathbf{x}_0$, we compute $\mathbf{S}_{i+1} = F(\mathbf{S}_i)$ for $i = 0, \ldots, 2k-1$ (the *basic iterations*), we apply the STEA1 or the STEA2 to these vectors or matrices, and we set $\mathbf{x}_1 = \widehat{\boldsymbol{\varepsilon}}_{2k}^{(0)}$ or $\mathbf{x}_1 = \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(0)}$. The whole process (denoted a *cycle*) is then restarted with $\mathbf{S}_0 = \mathbf{x}_1$. It is named the *Restarted Method* and denoted RM.

When, as particular case, we take $k = m$ (the dimension of the system) it was proved that the sequence $(\mathbf{x}_n)$ converges quadratically to the fixed point $\mathbf{S}$ of F under some assumptions [41]. This last method can be considered as a generalization to higher dimensions

$$
\begin{array}{ccccc}
\text{column } 0 & & \text{column } 2 & & \text{column } 4 \\
\boxed{\varepsilon_0^{(0)}} = \mathbf{S}_0 & & & & \\
\downarrow & & & & \\
\boxed{\varepsilon_0^{(1)}} = \mathbf{S}_1 & \longrightarrow & \boxed{\varepsilon_2^{(0)}} & & \\
 & & \downarrow & & \\
\varepsilon_0^{(2)} = \mathbf{S}_2 & & \boxed{\varepsilon_2^{(1)}} & \longrightarrow & \boxed{\varepsilon_4^{(0)}} \\
 & & & & \downarrow \\
\varepsilon_0^{(3)} = \mathbf{S}_3 & & \varepsilon_2^{(2)} & & \boxed{\varepsilon_4^{(1)}} \\
 & & & & \downarrow \\
\varepsilon_0^{(4)} = \mathbf{S}_4 & & \varepsilon_2^{(3)} & & \boxed{\varepsilon_4^{(2)}} \\
 & & & & \downarrow \\
\varepsilon_0^{(5)} = \mathbf{S}_5 & & \varepsilon_2^{(4)} & & \boxed{\varepsilon_4^{(3)}} \\
\vdots & & \vdots & & \vdots \\
\varepsilon_0^{(2k)} = \mathbf{S}_{2k} & & \varepsilon_2^{(2k-1)} & & \boxed{\varepsilon_4^{(2k-2)}} \\
\vdots & & \vdots & & \vdots
\end{array}
$$

TABLE 2. Values obtained with a maximal column $2k = 4$ fixed.

of Steffensen's method for $m = 1$ [63]. This is why it has been called the *Generalized Steffensen Method* and denoted by GSM. Quadratic convergence still occurs if k is the degree of the minimal polynomial of F' for the vector $\mathbf{S}_0 - \mathbf{S}$. This type of convergence has also been observed for much smaller values of k. Let us notice that each term of the sequence $(\mathbf{x}_n)$ needs $2k$ evaluations of the system of equations. Notice that, in our case, the system of nonlinear equations to be solved is given by (7), and, thus, $m = p + 1$.

The RM and the GSM can be extended to the case where $F : \mathbb{R}^{m\times s} \longmapsto \mathbb{R}^{m\times s}$, but the quadratic character of the convergence has yet to be studied.

The concept of *mesh independence principle* is of interest in our context. It states that, under reasonable assumptions on the numerical integration scheme, the iterates produced by a quasi-Newton method for solving (4) and (5) (which is the case of the RM as explained in [22]) behave asymptotically the same. Therefore, the cost in the number of iterates necessary to achieve a given precision is essentially the same for both systems [1, 39, 50, 68].

Let us mention that the choice of $\mathbf{y}$ is a difficult and unsolved problem in the general case. The experience acquired after many numerical experiments has shown us that the most appropriate choices seem to be either $\mathbf{y} = (1, \ldots, 1)^T$ or a random $\mathbf{y}$ [20, 21]. However, for some special classes of sequences, theoretical results have been obtained [19].

4. Examples. We will now give examples of the use of the STEA1 and the STEA2 for accelerating the iterations (7) or solving the system of nonlinear equations (5) coming out from our method for computing an approximate solution of Fredholm integral equations of the second kind.

For our examples, we remind that we used the trapezoidal rule (see, for example, [49]). We set $h = (b-a)/p$, $x_j^{(p)} = a + jh$ for $j = 0, \ldots, p$, and we have

$$\begin{aligned}\int_a^b K(t,x,u(x))\,dx \;\simeq\; & \frac{h}{2}\left[K(t,x_0^{(p)},u(x_0^{(p)})) + 2\sum_{j=1}^{p-1} K(t,x_j^{(p)},u(x_j^{(p)}))\right.\\ & \left. + K(t,x_p^{(p)},u(x_p^{(p)}))\right].\end{aligned}$$

Thus the system of $p+1$ nonlinear equations to be solved becomes

$$u_i = \frac{h}{2}\left[K(t_i,t_0,u_0) + 2\sum_{j=1}^{p-1} K(t_i,t_j,u_j) + K(t_i,t_p,u_p)\right] + f_i,$$

for $i = 0, \ldots, p$, and we solved it by the iterative procedure

$$\begin{aligned}u_i^{(n+1)} \;=\; & u_i^{(n)} - \alpha\left\{u_i^{(n)} - \frac{h}{2}\left[K(t_i,t_0,u_0^{(n)}) + 2\sum_{j=1}^{p-1} K(t_i,t_j,u_j^{(n)})\right.\right.\\ & \left.\left. + K(t_i,t_p,u_p^{(n)})\right] - f_i\right\}, \qquad i = 0, \ldots, p.\end{aligned}$$

These iterations can be written as

$$\mathbf{u}^{(n+1)} = F(\mathbf{u}^{(n)}).$$

The value of the parameter α is not the same for all examples. It was chosen to obtain convergence of the iterations.

All examples are started from $u_i^{(0)} = 1$ for $i = 0, \ldots, p$. The STEA1 and the STEA2 will be applied to the sequence of vectors $\mathbf{S}_n = \mathbf{u}^{(n)} = (u_0^{(n)}, \ldots, u_p^{(n)})^T$, and the scalar ε–algorithm to the sequence of scalars $s_n = (\mathbf{y}, \mathbf{u}^{(n)})$, where $(\cdot, \cdot)$ is the usual scalar product. In our examples, two possible choices are made for the vector $\mathbf{y}$: it is randomly chosen in $[-1, 1]$ (a choice which can lead to quite different results), or it is set to $\mathbf{y} = (1, \ldots, 1)^T$. Due to the homogeneity property of the scalar ε–algorithm, this second choice is equivalent to applying it to the sequence (s_n) whose terms are the mean values of the components of the vectors $\mathbf{S}_n$. In the examples, we will show the results obtained by the Acceleration Method (AM), and those that came from the Restarted Method (RM) and the Generalized Steffensen Method (GSM), that is when the RM is applied with $k = p + 1$. Sometimes, the value of k is not the same for the AM and for the RM. The reason is that one cycle of the RM needs $2k$ basic iterations thus the number of iterations is equal to $2k$ times the number of cycles, while the AM needs $2k$ basic iterations to reach the first term of the column $2k$, and one additional iteration for each new term in that column. Thus, the comparison between the AM and the RM for reaching a given precision has to take into account the total number of evaluations of F.

In some examples we plot the infinity norm of the error while, in others, it is the infinity norm of the difference, called the residual, between a result and F applied to it. The advantage of using the residual is to allow us to check the convergence to the solution of the discretized fixed point problem, which is not the exact solution of the integral equation, without knowing it. In the case of the AM we show the errors or the residuals of the basic iterations and of the extrapolated ones. For the RM, we show the errors or the residuals at the points $\mathbf{x}_n$ (indicated by a special character) and, in between, the errors or the residuals of the basic iterations. The fact that, in some cases, the residuals are much smaller than the errors means that the system of equations (obtained after discretization by the quadrature rule) has been well solved (the fixed point of F has been reached) but that, due to the error in the quadrature formula, its solution is not close to the exact solution of the integral equation.

As a stopping criterion for the use of the STEA1 or the STEA2 in the AM, we use the following inequalities on the residuals

$$\begin{array}{l} \|\widetilde{\varepsilon}_{2k}^{(n+1)} - F(\widetilde{\varepsilon}_{2k}^{(n+1)})\| > M\|\widetilde{\varepsilon}_{2k}^{(n)} - F(\widetilde{\varepsilon}_{2k}^{(n)})\| \quad \text{or} \\ \|\widehat{\varepsilon}_{2k}^{(n+1)} - F(\widehat{\varepsilon}_{2k}^{(n+1)})\| > M\|\widehat{\varepsilon}_{2k}^{(n)} - F(\widehat{\varepsilon}_{2k}^{(n)})\|, \end{array}$$

or

$$\|\widetilde{\varepsilon}_{2k}^{(n)} - F(\widetilde{\varepsilon}_{2k}^{(n)})\| \leq \tau \quad \text{or} \quad \|\widehat{\varepsilon}_{2k}^{(n)} - F(\widehat{\varepsilon}_{2k}^{(n)})\| \leq \tau,$$

where M and τ are chosen by the user, and F is as defined above. Let us comment on these tests. When convergence occurs with a precision close to machine's, oscillations arise due to instability and there is no advantage to continue the iterations. This is the role of the first line of tests. A large value of M (for example $M = 20$) will allow oscillations in the results, while the iterates will be stopped with a small one (for example $M = 2$). The tests in the second line are related to the convergence. A large value of τ (for example 10^{-4}) will stop the iterations too early, while a much smaller value will never stop the iterations (no convergence reached). For the RM, the stopping criterion we used is the number of cycles. Obviously, it can be easily replaced by a test on the error of the fixed point iterates.

In the Figures, the norm (in log scale) is the infinity one, but the choice of the Euclidean norm can be made in the software. In the legends, `eps` denotes the values obtained by the corresponding STEA algorithm, and `sol` is $\mathbf{u}$, the exact solution. For the AM, $\mathbf{u}^{(n)}$ is designated by `u`, while for the RM, it is designated by $\mathtt{u}_{\mathtt{ori}}$. In the RM, the `u`'s denote the basic iterates $\mathbf{u}^{(n)}$ obtained after restarting from the last extrapolated value.

The precision of the results cannot go beyond the error of quadrature rule used, that is, for the trapezoidal rule, $\|\mathbf{u}^{(n)} - \mathbf{u}\|_\infty = (b-a)\widetilde{M}h^2/12$ where $\widetilde{M} = \max_{t\in[a,b]} |K_x^{''}(t, x, u(x))|$, see [4, 5]. Obviously, using a more precise quadrature formula, such a Gaussian one, will produce better results. When full precision is achieved, stagnation occurs and the algorithm can stop due to a division by a number smaller than our tolerance.

The examples are taken from the literature on the topic, and, for each of them, we indicate the corresponding reference(s). The results were obtained with MATLAB R2010b. Differences can be observed with other versions of MATLAB and/or other processors. More examples are given in the software provided with this paper.

The RM has also been compared with Anderson acceleration and with the method of Lemaréchal [42], a method of dynamic Aitken-like relaxation which only requires two evaluations of F at each iteration. Anderson acceleration [2] is only a fixed point method for the solution of a system of nonlinear equations. It produces its own sequence and it cannot be used for accelerating a given sequence contrarily to the claims made in [55] and [24] which were based on a misunderstanding (on this topic, see [22]). Moreover, Anderson acceleration needs the solution, in the least squares sense, of a system of linear equations of increasing dimension at each iteration. For its implementation, we used the MATLAB program written by Walker [65] with the default parameters. The linear systems are solved by a QR decomposition, but, due to their increasing ill-conditioning, a dropping strategy of additional columns on the left has to be used to maintain an acceptable conditioning. Without it, the method diverges. In the corresponding Figures, the basic iterations of RM have not been plotted, Anderson acceleration is designated by AA, and Lemaréchal method by LM. It will be interesting to conduct more such experiments in a systematic way.

4.1. Example 1. This example is treated in [56]

$$u(t) = \frac{1}{5}\int_0^1 \cos(\pi t)\sin(\pi x)u^3(x)\,dx + \sin(\pi t).$$

Its solution is

$$u(x) = \sin(\pi x) + \frac{1}{3}(20 - \sqrt{391})\cos(\pi x).$$

The authors used a combination of Simpson method for evaluating the integral and the Newton–Kantorovich method for solving the system of nonlinear equations. They obtained a maximal error of 7.5×10^{-2} after one iteration of the Newton–Kantorovich method, 4.9×10^{-2} after 3 iterations, and then stagnate.

For the AM, we take $\alpha = 0.1, p = 21, 2k = 6$. With 50 iterations, we obtain the results on the left of Figure 1 for $\mathbf{y} = (1, \ldots, 1)^T$, and those on the right with a random $\mathbf{y}$. We see that, in this case, the

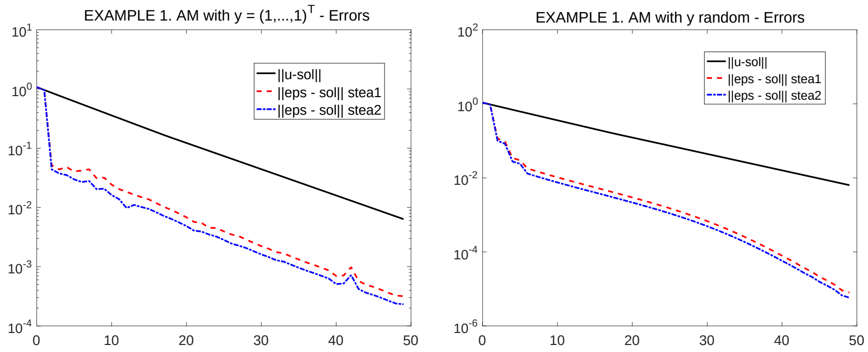

FIGURE 1. Example 1. AM with $\mathbf{y} = (1, \dots, 1)^T$ (left) and $\mathbf{y}$ random (right).

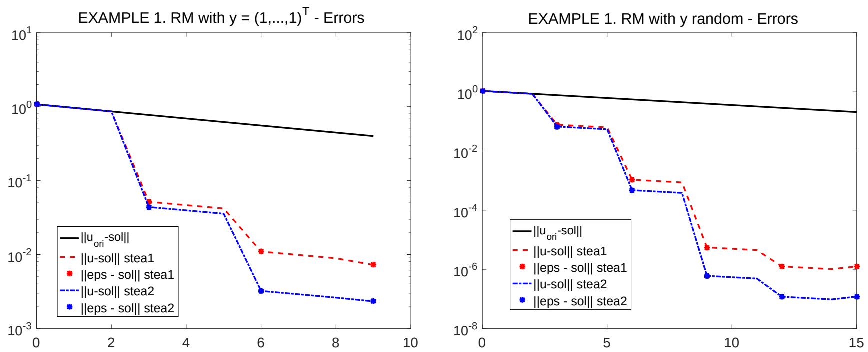

FIGURE 2. Example 1. RM with $\mathbf{y} = (1, \dots, 1)^T$ (left) and $\mathbf{y}$ random (right).

random $\mathbf{y}$ leads to much better results. The AM has been compared with the MPE, the MMPE, and the RRE in terms of level of accuracy reached with the same number of iterations. The results obtained are quite similar.

For the RM, with $\alpha = 0.1, p = 21$, and $2k = 2$, we get the results of Figure 2. With $\mathbf{y} = (1, \dots, 1)^T$, only 3 iterates of the RM are possible due to a division by a value smaller than the tolerance used in our toolbox. This is due to the stagnation of the RM which achieves its full accuracy, thus leading to the difference of two almost equal quantities in a denominator of the scalar ε–algorithm.

We compared RM (STEA2) with the other methods but with $\alpha = 0.01$ and again $2k = 2$. Each iteration of the RM and Lemaréchal method needs 2 evaluations of F. The RM again stagnated around 10^{-3} afer 2 iterations. The method of Lemaréchal reaches a precision of 10^{-8} after 5 iterations while Anderson acceleration goes to an error o 10^{-4} at iteration 5 (see left Figure 4). This example was also computed by the HAM in [34].

4.2. Example 2. This example was given by Anderson [2]

$$u(t) = \frac{3\sqrt{2}\pi}{16} \int_{-1}^{1} \cos(\pi|t-x|/4)u^2(x)\,dx - \cos(\pi t/4)/4.$$

Its solution is

$$u(x) = \cos(\pi x/4).$$

With $\alpha = 0.1$, the basic iterates strongly diverge and so the AM, as noticed by Anderson who made use of the vector ε–algorithm. The RM with $\alpha = 0.001, \mathbf{y} = (1,\ldots,1)^T$, $p = 21$ and $2k = 6$, gives the results on the left of Figure 3. On the right, we see the exact and the approximate solutions. This example is quite sensitive.

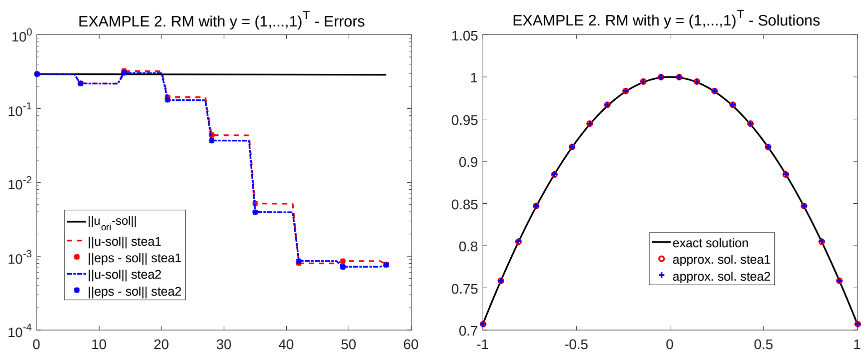

FIGURE 3. Example 2. RM with $\mathbf{y} = (1,\ldots,1)^T$ (left). Exact and approximate solutions (right).

In Figure 4, Anderson acceleration, the method of Lemaréchal and the RM (STEA2) have been compared (right).

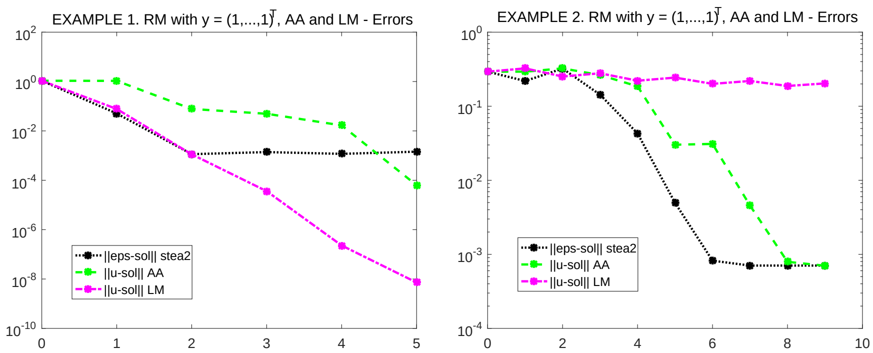

FIGURE 4. Comparison of the methods. Example 1 (left), Example 2 (right).

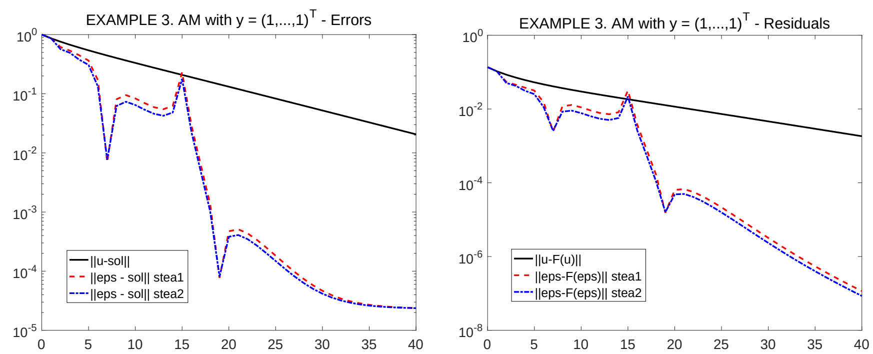

FIGURE 5. Example 3. AM with $\mathbf{y} = (1, \ldots, 1)^T$ errors (left), residuals (right).

4.3. Example 3. An error was corrected in the equation of example 5.3 considered in [8] ($+1$ instead of -1)

$$u(t) = -\int_0^1 (x+t)e^{u(x)}\,dx + et + 1,$$

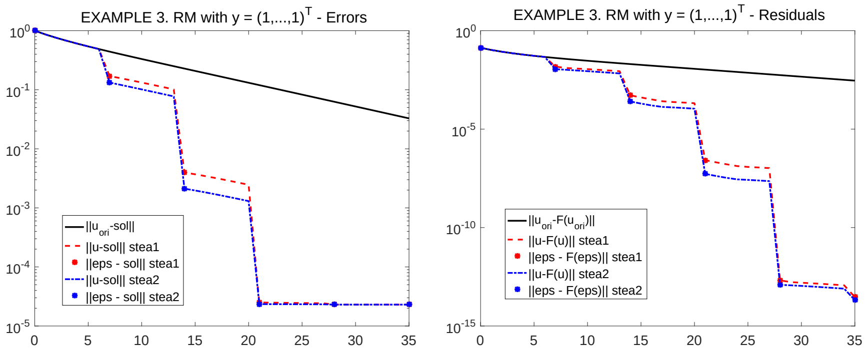

FIGURE 6. Example 3. RM with $\mathbf{y} = (1,\dots,1)^T$ errors (left), residuals (right).

whose solution is $u(x) = x$. The authors discretize the integral by a Newton–Cotes formula and then perform the basic iterative scheme (Picard iterations or the relaxation method). In $[0,1]$, the error is in the interval $[-5\times 10^{-3}, 10^{-2}]$.

With $\alpha = 0.1, p = 100, \mathbf{y} = (1,\dots,1)^T$, and $2k = 6$, the AM produces the error curves on the left of Figure 5 and the residuals on the right. We asked for 50 iterations but, as it can be seen, the iterations stopped at 40 because $\tau = 10^{-7}$ (and $M = 20$). With the same τ and $M = 2$, the iterations terminate at 8. With $p = 10$, the error after 40 iterations is 2.3×10^{-3} while it reaches 2.4×10^{-5} when $p = 100$ which confirms that the error of the trapezoidal rule behaves as $\mathcal{O}(h^2)$. The results obtained by the RM are given on Figure 6. The Figures 5 and 6 allow to compare the errors and the residuals, and they show a quite similar behavior. There is stagnation of the error of the quadrature formula because full accuracy has been obtained. The residuals do not stagnate because the fixed point of F for the STEAs has not been reached. Between the 4th and the 5th cycle the residuals decrease from 2.09×10^{-13} to 2.95×10^{-14} while the errors stagnate but around 10^{-4} which is a quite different level of precision. It not possible to perform an additional cycle because a division by a too small quantity arises. A comparison with other method can be seen in Figure 8 (left).

Table 3 shows the results obtained by the AM for various values of the parameters α, p and k after 25 iterations and with $\mathbf{y} = (1, \ldots, 1)^T$. With $\alpha = 0.2, p = 100, 2k = 6$ and a random vector $\mathbf{y}$, the errors are 2.30×10^{-5} for the STEA1 and the STEA2, and the residuals are 8.22×10^{-8} and 6.39×10^{-9} respectively.

α	p	$2k$	err STEA1	err STEA2	res STEA1	res STEA2	$\Vert\mathtt{u}-\mathtt{sol}\Vert$	$\Vert\mathtt{u}-\mathtt{F(u)}\Vert$
0.1	100	4	8.08×10^{-3}	6.83×10^{-3}	9.98×10^{-4}	7.82×10^{-4}	9.08×10^{-2}	8.02×10^{-3}
0.1	100	6	2.51×10^{-4}	2.02×10^{-4}	3.08×10^{-5}	2.24×10^{-5}	9.08×10^{-2}	8.02×10^{-4}
0.2	100	2	1.13×10^{-4}	9.85×10^{-5}	1.99×10^{-5}	1.48×10^{-5}	7.80×10^{-3}	1.39×10^{-3}
0.2	100	4	2.34×10^{-5}	2.33×10^{-5}	1.07×10^{-7}	6.59×10^{-8}	7.80×10^{-3}	1.39×10^{-3}
0.2	100	6	2.11×10^{-5}	2.18×10^{-5}	5.43×10^{-7}	2.89×10^{-7}	7.80×10^{-3}	1.39×10^{-3}
0.2	100	10	2.30×10^{-5}	2.30×10^{-5}	1.08×10^{-10}	4.07×10^{-11}	7.80×10^{-3}	1.39×10^{-3}
0.5	100	6	2.30×10^{-5}	2.30×10^{-5}	5.41×10^{-11}	6.76×10^{-12}	2.22×10^{-5}	4.87×10^{-7}
0.1	10	6	2.53×10^{-3}	2.48×10^{-3}	3.14×10^{-5}	2.29×10^{-5}	8.94×10^{-2}	8.07×10^{-3}
0.2	10	6	2.30×10^{-3}	2.30×10^{-3}	3.56×10^{-7}	1.89×10^{-7}	5.64×10^{-3}	1.41×10^{-3}
0.5	10	6	2.30×10^{-3}	2.30×10^{-3}	6.03×10^{-11}	7.49×10^{-12}	2.30×10^{-3}	5.28×10^{-7}

TABLE 3. Example 3. Results with $\mathbf{y} = (1, \ldots, 1)^T$ and various choices of the parameters.

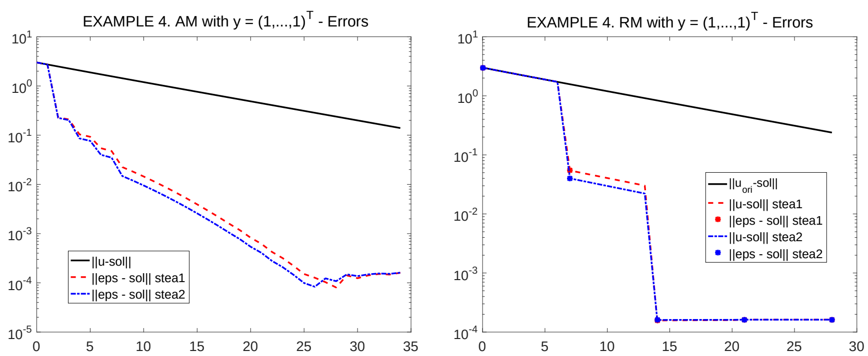

FIGURE 7. Example 4. AM (left) and RM (right) with $\mathbf{y} = (1, \ldots, 1)^T$.

4.4. Example 4. In reference [43], we found the equation (Ex. 1)

$$u(t) = t/20 \int_0^1 xu^2(x)\,dx + 3 + 0.6625t.$$

Its solution is $u(x) = x + 3$. The computation used Haar wavelets. With 16 of them, they obtained an error similar to our.

We took $\alpha = 0.1, p = 21$, and $\mathbf{y} = (1, \dots, 1)^T$. With $2k = 8$, the AM gives the results on the left of Figure 7, while the RM with $2k = 6$ gives the curves on the right. We see that quadratic convergence has been achieved by RM with k much smaller than m.

In Figure 8, Anderson acceleration, the method of Lemaréchal and the RM (STEA2) are compared (right).

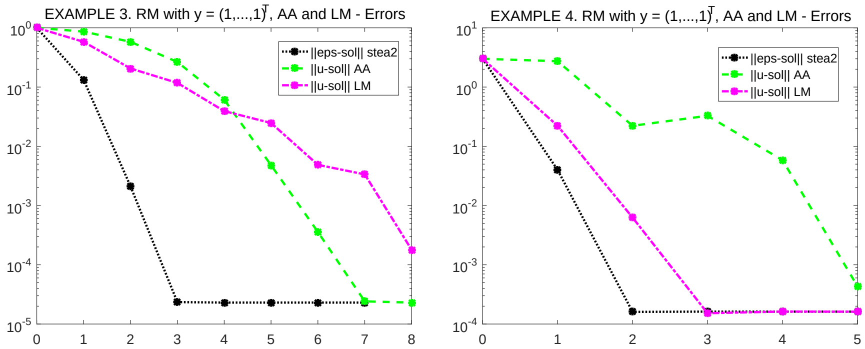

FIGURE 8. Comparison of the methods. Example 3 (left), Example 4 (right).

4.5. Example 5. Consider now

$$u(t) = t^2 \int_0^1 \frac{x^2}{1 + u^2(x)}\, dx + (1/2 - \ln 2)t^2 + \sqrt{t},$$

whose solution is $u(x) = \sqrt{x}$ [43] (Ex. 4). The results obtained is comparable to our.

The AM with $\alpha = 0.2, p = 31, \mathbf{y} = (1, \dots, 1)^T$, and $2k = 10$ give the results on the left of Figure 9. With $2k = 4$, the RM gives the results on the right of this Figure.

4.6. Example 6. Let us end by an example of a linear Fredholm equation [44] (Ex. 1)

$$u(t) = \int_0^1 \sin(4\pi t + 2\pi x) u(x)\, dx + \cos(2\pi t) + \sin(4\pi t)/2$$

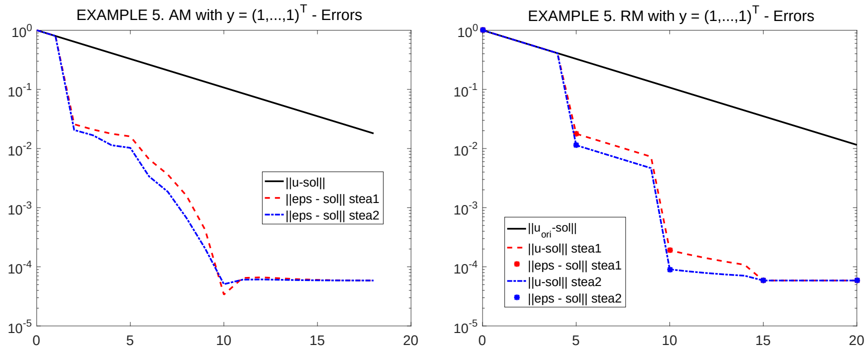

FIGURE 9. Example 5. AM (left) and RM (right) with $\mathbf{y} = (1, \dots, 1)^T$.

whose solution is $u(x) = \cos(2\pi x)$. Take $\alpha = 0.1, p = 3, \mathbf{y} = (1, \dots, 1)^T$ and $\tau = 10^{-9}$. Thus, there are only 4 nodes. With $2k = 8$, the AM and the RM (which, in this case, is the GSM) gives the results of Figure 10. The theory of Shanks transformations tells us that the exact solution of a system of linear equations is obtained in one iteration when $2k = 2(p + 1)$.

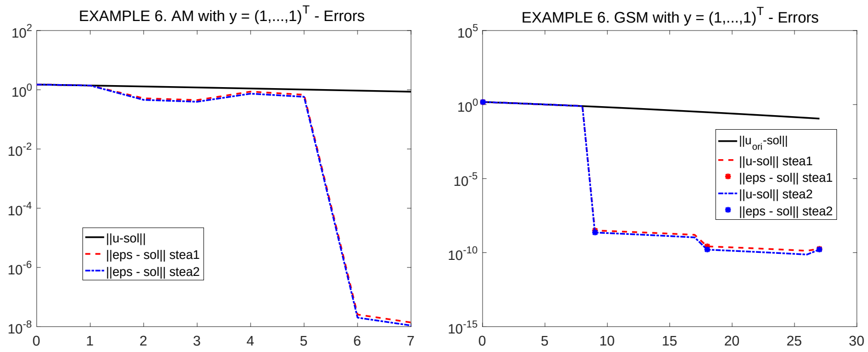

FIGURE 10. Example 6. AM (left) and GSM (right) with $\mathbf{y} = (1, \dots, 1)^T$.

In Figure 11, we show the results obtained by Anderson acceleration, the method of Lemaréchal and the RM on Example 5 (left) and on Example 6 (right). Example 6 was obtained by the STEA1 but now with $p = 10$ and $2k = 2$ instead of $p = 3$) and $2k = 8$. Thus the RM is no longer the GSM, and it does not reach the exact solution. The RM and the method of Lemaréchal need 2 evaluations of F per iteration. The STEA2 stopped before due to a division by zero.

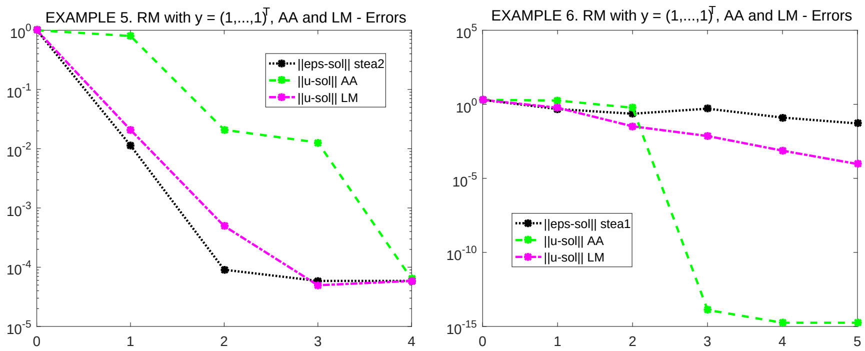

FIGURE 11. Comparison of the methods. Example 5 (left), Example 6 (right).

5. Conclusions. The aim of this paper was not to present a new method for solving nonlinear Fredholm equations of the second kind which provides better results than the methods existing in the literature in all situations. We only want to exemplify the fact that the Simplified Topological Epsilon Algorithms can accelerate the convergence of the Picard iterations (or, more generally, the relaxation method) for solving the system of nonlinear equations obtained after discretization of the integral by a quadrature formula (Acceleration Method), and to show that these iterations can be coupled with our algorithms by a restarting procedure (Restarted Method). The examples given above show that the RM (and still more the GSM) seems to be more effective than the AM but one has to remind that one step of the RM is costly in terms of evaluations of F, and, thus, the whole process is much more expensive than the AM. It must be noticed that, although quadratic convergence of the restarting process has only been proved in the case of the GSM, it also sometimes occurs with the RM. As seen from examples, the results obtained are quite sensitive to the choice of the parameter α which, maybe, could be advantageously replaced by a sequence (α_n) appropriately chosen. They also depend on the choice of $\mathbf{y}$, an important unsolved problem. A more precise quadrature formula than the trapezoidal rule could lead to better results. Our methods can also be used for Fredholm equations of the first kind and for Volterra equations. Since our software for the STEA1 and the STEA2 can accelerate matrix sequences and solve matrix equations, it can treat integral equations in two variables (see, for example, [53]).

Acknowledgments. We would like to thank Prof. K.E. Atkinson for his comments on an earlier version of this work. The remarks of the reviewers helped us to clarify many points of the paper and to make it more readable. We thank them.

REFERENCES

1. E. Allgower, K. B Öhmer, F. Potra, W. Rheinboldt, A mesh independence principle for operator equations and their discretizations, SIAM J. Num. Anal., 23 (1986) 160–169.

2. D.G. Anderson, Iterative procedures for nonlinear integral equations, J. Assoc. Comput. Mach., 12 (1965) 547–560.

3. K.E. Atkinson, An automatic program for linear Fredholm integral equations of the second kind, ACM Trans. Math. Soft., 2 (1976) 154–171.

4. ———, A survey of numerical methods for solving nonlinear integral equation, J. Integral Equations Appl., 4 (1992) 15–46.

5. K.E. Atkinson, F. Potra, The discrete Galerkin method for nonlinear integral equations, J. Integral Equations Appl., 1 (1988) 17–54.

6. K.E. Atkinson, L. Shampine. Algorithm 876: Solving Fredholm integral equations of the second kind in Matlab, ACM Trans. Math. Software, 34 (2008), Article 21, DOI: 10.1145/1377596.1377601.

7. V. Babenko, Numerical methods for Volterra and Fredholm integral equations for functions with values in L-spaces, Appl. Math. Comput., 291 (2016) 354–372.

8. A.H. Borzabadi, O.S. Fard, A numerical scheme for a class of nonlinear Fredholm integral equations of the second kind, J. Comput. Appl. Math., 232 (2009) 449–454.

9. C. Brezinski, Application de l'ε–algorithme à la résolution des systèmes non linéaires, C. R. Acad. Sci. Paris, 271 A (1970) 1174–1177.

10. ———, Sur un algorithme de résolution des systèmes non linéaires, C. R. Acad. Sci. Paris, 272 A (1971) 145–148.

11. ———, Généralisation de la transformation de Shanks, de la table de Padé et de l'ε–algorithme, Calcolo, 12 (1975) 317–360.

12. ———, Numerical stability of a quadratic method for solving systems of nonlinear equations, Computing, 14 (1975) 205–211.

13. ———, *Projection Methods for Systems of Equations* North-Holland, Amsterdam, 1997.

14. ———, Extrapolation algorithms for filtering series of functions, and treating the Gibbs phenomenon, Numer. Algorithms, 36 (2004) 309–329.

15. C. Brezinski, J.–P. Chehab, Nonlinear hybrid procedures and fixed point iterations, Numer. Funct. Anal. Optimization, 19 (1998) 465–487.

16. C. Brezinski, M. Crouzeix, Remarques sur le procédé Δ^2 d'Aitken, C. R. Acad. Sci. Paris, 270 A (1970) 896–898.

17. C. Brezinski, M. Redivo-Zaglia, *Extrapolation Methods. Theory and Practice*, North-Holland, Amsterdam, 1991.

18. ———, Rational extrapolation of the PageRank vectors, Math. Comp., 77 (2008) 1585–1598.

19. ———, The simplified topological ε–algorithms for accelerating sequences in a vector space, SIAM J. Sci. Comput., 36 (2014) A2227–A2247.

20. ———, Convergence acceleration of Kaczmarz's method, J. Engrg. Math., 93 (2015) 3–19.

21. ———, The simplified topological ε–algorithms: software and applications, Numer. Algorithms, 74 (2017) 1237–1260.

22. C. Brezinski, M. Redivo–Zaglia, Y. Saad, Shanks sequence transformations and Anderson acceleration, SIAM Rev., to appear.

23. S. Cabay, L.W. Jackson, A polynomial extrapolation method for finding limits and antilimits of vector sequences, SIAM J. Numer. Anal., 13 (1976) 734–752.

24. S.R. Capehart, *Techniques for Accelerating Iterative Methods for the Solution of Mathematical Problems*, Ph.D. Thesis, Oklahoma State University, Stillwater, Oklahoma, USA, July 1989.

25. J. Degroote, K. J. Bathe, J. Vierendeels, Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction, Computers and Structures, 87 (2009) 793–801.

26. D.R. Dellwo, Accelerated refinement with applications to integral equations, SIAM J. Numer. Anal., 25 (1988) 1327–1339.

27. R.P. Eddy, Extrapolation to the limit of a vector sequence, in *Information Linkage between Applied Mathematics and Industry*, P.C.C. Wang ed., Academic Press, New York, 1979, pp. 387–396.

28. P. Erbts, A. Düster, Accelerated staggered coupling schemes for problems of thermoelasticity at finite strains, Comput. Math. Appl., 64 (2012) 2408–2430.

29. E. Gekeler, On the solution of systems of equations by the epsilon algorithm of Wynn, Math. of Comp., 26 (1972) 427–436.

30. B. Germain-Bonne, *Estimation de la Limite de Suites et Formalisation de Procédés d'Accélération de Convergence*, Thèse de Doctorat d'État, Université des Sciences et Techniques de Lille, 1978.

31. P.R. Graves–Morris, Solution of integral equations using generalised inverse, function-valued Padé approximants, I, J. Comput. Appl. Math., 32 (1990) 117–124.

32. ———, Solution of integral equations using function-valued Padé approximants, II, Numer. Algorithms, 3 (1992) 223–234.

33. P.R. Graves–Morris, D.E. Roberts, A. Salam, The epsilon algorithm and related topics, J. Comput. Appl. Math., 122 (2000) 51–80.

34. A. Jafarian, Z. Esmailzadeh, L. Khoshbakhti, A numerical method for solving nonlinear integral equations in the Urysohn form, Appl. Math. Sci., 7 (2013) 1375–1385.

35. K. Jbilou, A general projection algorithm for solving systems of linear equations, Numer. Algorithms, 4 (1993) 361–377.

36. K. Jbilou, H. Sadok, Some results about vector extrapolation methods and related fixed point iteration, J. Comp. Appl. Math., 36 (1991) 385–398.

37. ———, Matrix polynomial and epsilon–type extrapolation methods with applications, Numer. Algorithms, 68 (2015) 107–119.

38. K. Kalbasi, K. Demarest, Convergence acceleration techniques in the iterative solution of integral equation problems, in *Antennas and Propagation Society International Symposium, 1990. AP-S. Merging Technologies for the 90's. Digest*, IEEExplore, 1990, pp. 64–67.

39. C.T. Kelley, E.W. Sachs, Mesh independence of Newton–like methods for infinite dimensional problems, J. Integral Equations Appl., 3 (1991) 549–573.

40. U. Küttler, W.A. Wall, Fixed-point fluid-structure interaction solvers with dynamic relaxation, Comput. Mech., 43 (2008) 61–72.

41. H. Le Ferrand, The quadratic convergence of the topological epsilon algorithm for systems of nonlinear equations, Numer. Algorithms, 3 (1992) 273–284.

42. C. Lemaréchal, Une méthode de résolution de certains systèmes non linéaires bien posés, C.R. Acad. Sci. Paris, sér. A, 272 (1971) 605–607.

43. U. Lepik, E. Tamme, Solution of nonlinear Fredholm integral equations via the Haar wavelet method, Proc. Estonian Acad. Sci. Phys. Math., 56 (2007) 17–27.

44. X.–Z. Liang, M.–C. Liu, X.–J. Che, Solving second kind integral equations by Galerkin methods with continuous orthogonal wavelets, J. Comput. Appl. Math., 136 (2001) 149–161.

45. W.R. Mann, Mean value methods in iteration, Proc. Amer. Math. Soc., 4 (1953) 506–510.

46. A.J. MacLeod, Acceleration of vector sequences by multi-dimensional Δ^2 methods, Comm. Appl. Numer. Meth., 2 (1986) 385-392.

47. A.C. Matos, Convergence and acceleration properties for the vector ε–algorithm, Numer. Algorithms, 3 (1992) 313–320.

48. M. Mešina, Convergence acceleration for the iterative solution of $x = Ax + f$, Comput. Methods Appl. Mech. Eng., 10 (1977) 165–173.

49. S. Micula, On some numerical iterative methods for Fredholm integral equations with deviating arguments, Stud. Univ. Babeş-Bolyai Math., 61 (2016) 331–341.

50. I. Moret, P. Omari, Iterative solution of integral equations by a quasi-Newton method. J. Comput. Appl. Math., 20 (1987) 333–340.

51. Y. Nievergelt, Aitken's and Steffensen's accelerations in several variables, Numer. Math., 59 (1991) 295-310.

52. J.M. Ortega, W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

53. B.G. Pachpatte, Integral equations in two variables, in *Multidimensional Integral Equations and Inequalities*, Atlantis Studies in Mathematics for Engineering and Science, vol. 9, Atlantis Press, Amsterdam.

54. B.P. Pugachev, Acceleration of convergence of iterative processes and a method of solving systems of non–linear equations, USSR Comput. Maths. Maths. Phys., 17 (5) (1978) 199–207.

55. I. Ramière, T. Helfer, Iterative residual-based vector methods to accelerate fixed point iterations, Comput. Math. Appl., 70 (2015) 2210–2226.

56. J. Saberi-Nadjafi, M. Heidari, Solving nonlinear integral equations in the Urysohn form by Newton Kantorovich quadrature method, Comput. Math. Appl., 60 (2010) 2058–2065.

57. G.A. Sedogbo, Some convergence acceleration processes for a class of vector sequences, Applicationes Mathematicae, 24 (3) (1997) 299-306.

58. D. Shanks, Non linear transformations of divergent and slowly convergent sequences, J. Math. Phys., 34 (1955) 1–42.

59. A. Sidi, Extrapolation vs. projection methods for linear systems of equations, J. Comp. Appl. Math., 22 (1988) 71–88.

60. ———, *Practical Extrapolation Methods. Theory and Applications*, Cambridge University Press, Cambridge, 2003.

61. A. Sidi, W.F. Ford, D.A. Smith, Acceleration of convergence of vector sequences, SIAM J. Numer. Anal., 23 (1986) 178–196.

62. D.A. Smith, W.F. Ford, A. Sidi, Extrapolation methods for vector sequences, SIAM Rev., 29 (1987) 199–233.

63. J.F. Steffensen, Remarks on iteration, Skand. Aktuarietidskr., 16 (1933) 64-72.

64. R. Thukral, Solution of integral equations using Padé type approximants, J. Integral Equations Appl., 13 (2001) 181–206.

65. H.F. Walker, Anderson acceleration: algorithms and implementations, Worcester Polytechnic Institute, Mathematical Sciences Department, Research Report MS-6-15-50, June 2011.

66. H.F. Walker, P. Ni, Anderson acceleration for fixed-point iterations, SIAM J. Numer. Anal., 49 (2011) 1715–1735.

67. A.-M. Wazwaz, Applications of integral equations, in *Linear and Nonlinear Integral Equations*, Springer, Berlin, Heidelberg, 2011, pp. 569–595.

68. M. Weiser, A. Schiela, P. Deuflard, Asymptotic mesh independence of Newton's method revisited, SIAM J. Numer. Anal., 42 (2005) 1830–1845.

69. E.J. Weniger, Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series, Comp. Phys. Reports, 10 (1989) 189–371.

70. J. Wimp, *Sequence Transformations and their Applications*, Academic Press, New York, 1981.

71. P. Wynn, On a device for computing the $e_m(S_n)$ transformation, MTAC, 10 (1956) 91–96.

72. ———, Acceleration techniques for iterated vector and matrix problems, Math. Comp., 16 (1962) 301–322.

73. ———, An arsenal of Algol procedures for the evaluation of continued fractions and for effecting the epsilon algorithm, Chiffres, 4 (1966) 327–362.

74. S.M. Zemyan, *The Classical Theory of Integral Equations. A Concise Treatment*, Birkhäuser, Basel, 2012.

Laboratoire Paul Painlevé, UMR CNRS 8524, UFR de Mathématiques, Université de Lille - Sciences et Techniques, 59655–Villeneuve d'Ascq cedex, France
Email address: **claude.brezinski@univ-lille1.fr**

Università degli Studi di Padova, Dipartimento di Matematica "Tullio Levi-Civita", Via Trieste 63, 35121–Padova, Italy.
Email address: **Michela.RedivoZaglia@unipd.it**