Iranian Journal of Numerical Analysis and Optimization
Vol. 15, No. 1, 2025, pp 54–78
https://doi.org/10.22067/ijnao.2024.85903.1361
https://ijnao.um.ac.ir/

Research Article

An approach of extrapolation methods for the solution of nonlinear Volterra–Fredholm integral equations of the second kind

H. Safdari, M. Moradkhani* and M.T. Dastjerdi

Abstract

This study presents the process of using extrapolation methods to solve the nonlinear Volterra–Fredholm integral equations of the second kind. To do this, by approximating the integral terms contained in equations by a quadrature rule, the nonlinear Volterra–Fredholm integral equations of the

*Corresponding author

Received 16 December 2023; revised 6 August 2024; accepted 21 September 2024

Hamid Safdari
Faculty of Sciences, Teacher Training Shahid Rajaei University, Lavizan, Tehran, Iran. e-mail: hsafdari@sru.ac.ir

Maryam Moradkhani
Faculty of Sciences, Teacher Training Shahid Rajaei University, Lavizan, Tehran, Iran. e-mail: m.moradkhani@sru.ac.ir

Mohammad Taghi Dastjerdi
Faculty of Mathematics, Zanjan University, University Boulevard, Zanjan, Iran. e-mail: tdast@znu.ac.ir

How to cite this article
Safdari, H., Moradkhani, M. and Dastjerdi, M.T., An approach of extrapolation methods for the solution of nonlinear Volterra–Fredholm integral equations of the second kind. *Iran. J. Numer. Anal. Optim.*, 2025; 15(1): 54-78. https://doi.org/10.22067/ijnao.2024.85903.1361

second kind are reduced to a set of nonlinear algebraic equations. Then, the solution of the corresponding system of nonlinear equations is approximated by an iterative method, and finally, these iterations are accelerated by an extrapolation method. We demonstrate the effectiveness of the proposed approach by solving some numerical examples.

AMS subject classifications (2020): Primary 45G10; Secondary 65B05, 65B99.

Keywords: Volterra–Fredholm integral equations; Extrapolation methods; Iterative methods.

1 Introduction

The modeling of many mathematical, physics, biological, and engineering problems has led to the formation of a group of integral equations in the form of Volterra–Fredholm integral equations [30, 12]. Our focus is on the nonlinear Volterra–Fredholm integral equation of the second kind, represented as

$$x(t) = f(t) + \int_a^t g(t, s, x(s))ds + \int_a^b k(t, s, x(s))ds, \qquad t \in [a, b], \qquad (1)$$

where f, g, and k are known and analytic functions, and x is the unknown function to be determined. The existence, uniqueness, and other properties of the solution of equation (1) are presented in [24].

Since these equations cannot be solved exactly, it is important to find their approximate solutions using some numerical methods. There have been various suggested numerical methods for approximating solutions to nonlinear Volterra–Fredholm integral equations, such as radial basis functions [17], combination of the modified Adomian decomposition and method and the quadrature (trapezoidal and Weddle) rules [22], approximation collocation methods [11], Taylor polynomial methods [33], triangular functions methods [19], homotopy perturbation method [13], rationalized Haar functions methods [23, 1], wavelet methods [34], modification of hat functions [21], and many other methods.

In various science and engineering fields, it is essential to determine the limits or approximations of limits for sequences. For instance, approximations derived from fixed-point iterations of linear or nonlinear equation systems. Unfortunately, many of these sequences have slow convergence rates, making them expensive to use.

An effective solution to overcome this problem is to use extrapolation methods (or convergence acceleration methods) for the desired sequence. These methods use the original sequence to produce a new sequence that converges to the same limit as the original one but faster [29]. There exist many methods for accelerating the convergence of a sequence, such as the scalar Shanks transformation for transforming a sequence of numbers, which was introduced by Shanks [28], and Wynn [31] implemented it using the scalar ε-algorithm (SEA).

In 1962, Wynn [32] generalized the SEA to accelerate sequences of vectors. The vector ε-algorithm (VEA) was introduced using an algebraic theory similar to that of the scalar method. Brezinski extended the SEA to a topological vector space and created the Shanks topological transformation, which is implemented through a recursive algorithm in two classes [4]. The rules of these algorithms were later simplified and named the first simplified topological ε-algorithm (STEA1) and the second simplified topological ε-algorithm (STEA2) [8]. The topological Shanks transformation and the topological ε-algorithm (TEA) have been used for solving linear and nonlinear equations, computing eigenelements, Pade-type approximations, matrix functions, matrix equations, the Lanczo method, and more [9, 15, 5, 14, 25, 10].

Our study aims to show how we can use extrapolation methods to solve Volterra–Fredholm integral equations. Because these methods are defined without needing specific information about the sequence's generation, they can be directly utilized to solve linear and nonlinear systems. In this article, we find an iterative method to solve the system of nonlinear equations that result from using a quadrature method on the integral expressions of the Volterra–Fredholm integral equation. Then, we use extrapolation to accelerate the convergence of the sequence resulting from the iterative method and get an approximate solution to the integral equation. Section 2 presents a review of ε-algorithms and introduces the sequence transformations em-

ployed to accelerate the sequence produced in Section 3. In Section 3, we approximate the integral utilizing a quadrature method and then utilize the Picard iterative method to approximate it again. Additionally, we introduce an algorithm for accelerating the sequence produced by the iterative method. In Section 4, some numerical examples prove the efficacy of our method.

2 A review of ε-algorithms

2.1 The SEA

The Shanks transformation, proposed by Shanks in 1955, utilizes a recursive method to transform scalar sequences. If (x_n) is a scalar sequence and x is its unknown limit, applying the Shanks transformation converts the sequence into a set of sequences $(x_n) \to \{(e_k(x_n))\}$ that can be represented as a ratio of two determinants,

$$e_k(x_n) = \frac{\begin{vmatrix} x_n & \dots & x_{n+k} \\ \Delta x_n & \dots & \Delta x_{n+k} \\ \vdots & & \vdots \\ \Delta x_{n+k-1} & \dots & \Delta x_{n+2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ \Delta x_n & \dots & \Delta x_{n+k} \\ \vdots & & \vdots \\ \Delta x_{n+k-1} & \dots & \Delta x_{n+2k-1} \end{vmatrix}}, \qquad k, n = 0, 1, \dots.$$

In fact, for $k = 1$, Shanks transformation is the same as the Aitken's Δ^2 process.

Theorem 1. For all n, $e_k(x_n) = x$ if and only if there exist $a_0, \dots, a_k$, with $a_0 a_k \neq 0$ and $a_0 + \dots + a_k \neq 0$, such that

$$a_0(x_n - x) + \dots + a_k(x_{n+k} - x) = 0.$$

Proof. See [7]. □

The Shanks transformation can be performed recursively by Wynn's SEA [31], whose rules are

$$\begin{cases} \varepsilon_{-1}^{(n)} = 0, \quad n = 0, 1, \ldots, \\ \varepsilon_{0}^{(n)} = x_n, \quad n = 0, 1, \ldots, \\ \varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + (\varepsilon_{k}^{(n+1)} - \varepsilon_{k}^{(n)})^{-1}, \quad k, n = 0, 1, \ldots. \end{cases} \tag{2}$$

Wynn has shown that for all k and n, $\varepsilon_{2k}^{(n)} = e_k(x_n)$ and $\varepsilon_{2k+1}^{(n)} = 1/e_k(\Delta x_n)$, where the forward difference operator Δ is defined by $\Delta x_n = x_{n+1} - x_n$ [29].

The quantities $\varepsilon_k^{(n)}$ are usually displayed in a two-dimensional array known as the ε-array (Table 1).

Table 1: ε-array

$\varepsilon_{-1}^{(0)} = 0$					
	$\varepsilon_{0}^{(0)} = x_0$				
$\varepsilon_{-1}^{(1)} = 0$		$\varepsilon_{1}^{(0)}$			
	$\varepsilon_{0}^{(1)} = x_1$		$\varepsilon_{2}^{(0)}$		
$\varepsilon_{-1}^{(2)} = 0$		$\varepsilon_{1}^{(1)}$		$\ddots$	
$\vdots$	$\varepsilon_{0}^{(2)} = x_2$	$\vdots$			
$\vdots$	$\vdots$	$\vdots$			
$\vdots$	$\vdots$	$\vdots$			$\varepsilon_{2k}^{(0)}$
$\vdots$	$\vdots$	$\vdots$			
$\vdots$	$\vdots$	$\vdots$			
$\vdots$	$\vdots$	$\vdots$			
$\vdots$	$\varepsilon_{0}^{(2k-1)} = x_{2k-1}$	$\vdots$			
$\varepsilon_{-1}^{(2k)} = 0$		$\varepsilon_{1}^{(2k-1)}$			
	$\varepsilon_{0}^{(2k)} = x_{2k}$				

Only the values in the even columns, that is, the columns containing quantities with an even lower index, are interesting and directly related to the scalar Shanks transformation $\varepsilon_{2k}^{(n)} = e_k(x_n)$.

The values of the initial scalar sequence are saved in column 0. Thus, having $2k + 1$ terms of a sequence in column 0, that is, $\varepsilon_0^{(i)} = x_i$ for $i =$

$0, \ldots, 2k$, we are able to complete the ε-array up to the vertex $\varepsilon_{2k}^{(0)}$. With one difference in the calculation of the elements of column 1, we use this formula $\varepsilon_1^{(n)} = (\varepsilon_0^{(n+1)} - \varepsilon_0^{(n)})^{-1}$.

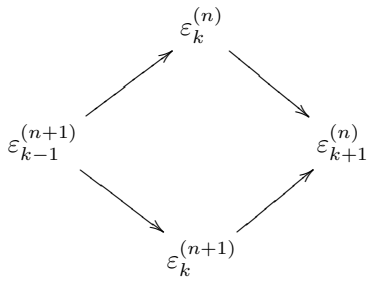
When dealing with a large initial sequence, it may be recommended to stop the computations before completing the entire ε-array up to its vertex, and once a certain even column (Maxcol) is reached (Table 2).

Table 2: ε-array with a certain even column

$\varepsilon_{-1}^{(0)} = 0$	$\varepsilon_0^{(0)} = x_0$			
$\varepsilon_{-1}^{(1)} = 0$	$\varepsilon_0^{(1)} = x_1$	$\varepsilon_1^{(0)}$	$\varepsilon_2^{(0)}$	
$\varepsilon_{-1}^{(2)} = 0$	$\varepsilon_0^{(2)} = x_2$	$\varepsilon_1^{(1)}$	$\ddots$	$\varepsilon_{Maxcol}^{(0)}$
	$\varepsilon_0^{(2k-1)} = x_{2k-1}$			$\varepsilon_{Maxcol}^{(1)}$
$\varepsilon_{-1}^{(2k)} = 0$	$\varepsilon_0^{(2k)} = x_{2k}$	$\varepsilon_1^{(2k-1)}$		$\varepsilon_{Maxcol}^{(2)}$
				$\varepsilon_{Maxcol}^{(3)}$
				$\varepsilon_{Maxcol}^{(4)}$

Although the easiest way to implement the ε-algorithm is to calculate the columns one by one from the first column, in this case, if we want to add a new term of the scalar sequence, all the calculations must be done again, which causes problems in computation time and storage space. To overcome these problems, Wynn proposed a method called moving rhombus [31] because, as we can see in (2), the expressions involved in calculation $\varepsilon_{k+1}^{(n)}$ are located in the four corners of a rhombus as Table 3.

Table 3: Rhombus rule



In this method, we proceed with ascending diagonals. Each diagonal will be a scalar vector, and we utilize the previous diagonal to calculate the next diagonal (Table 4).

Table 4: ε-array with a certain even column

		column 1	column 2	column 3	column 4
diagonal 1	$\boxed{\varepsilon_0^{(0)}} = x_0$				
	$\downarrow$	$\varepsilon_1^{(0)}$			
diagonal 2	$\boxed{\varepsilon_0^{(1)}} = x_1$	$\rightarrow$	$\boxed{\varepsilon_2^{(0)}}$		
		$\varepsilon_1^{(1)}$	$\downarrow$	$\varepsilon_3^{(0)}$	
diagonal 3	$\varepsilon_0^{(2)} = x_2$		$\boxed{\varepsilon_2^{(1)}}$	$\rightarrow$	$\boxed{\varepsilon_4^{(0)}}$
		$\varepsilon_1^{(2)}$		$\varepsilon_3^{(1)}$	$\downarrow$
diagonal 4	$\varepsilon_0^{(3)} = x_3$		$\varepsilon_2^{(2)}$		$\boxed{\varepsilon_4^{(1)}}$
		$\varepsilon_1^{(3)}$		$\varepsilon_3^{(2)}$	$\downarrow$
diagonal 5	$\varepsilon_0^{(4)} = x_4$		$\varepsilon_2^{(3)}$		$\boxed{\varepsilon_4^{(2)}}$
		$\varepsilon_1^{(4)}$		$\varepsilon_3^{(3)}$	$\downarrow$
diagonal 6	$\varepsilon_0^{(5)} = x_5$		$\varepsilon_2^{(4)}$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

For example, in ascending diagonal 2, quantities $\varepsilon_0^{(1)}$ and $\varepsilon_1^{(0)}$ have already been calculated and stored. In the calculation of the ascending diagonal 3, by entering the quantity $\varepsilon_0^{(2)} = x_2$ from the original scalar sequence and using the diagonal 2, we calculate the quantities $\varepsilon_1^{(1)}$ and $\varepsilon_2^{(0)}$. Now, diagonal 3 containing quantities $\varepsilon_0^{(2)}$, $\varepsilon_1^{(1)}$, and $\varepsilon_2^{(0)}$ is replaced the diagonal 2, and this process continues.

2.2 The VEA

When the elements of the sequence (x_n) are vectors, the VEA can be utilized. This algorithm is an extension of the SEA and is designed for vector sequences [32]. If we define the inverse of a vector x as $x^{-1} = x/(x, x)$, where $(\cdot, \cdot)$ represents the typical inner product, the rules for the vector algorithm are identical to those of the scalar algorithm. Additionally, the scheme and implementation are quite similar.

2.3 The TEA

In 1975, Brezinski [4] introduced the topological Shanks transformation, which is a more comprehensive approach as it deals with sequences of elements from a topological vector space E on $\mathbb{R}$ or $\mathbb{C}$. This transformation has two classes and two different algorithms for implementation. The idea was based on the definition of the inverse of a couple $(x, y) \in E \times E^*$ defined as $x^{-1} = y/\langle y, x\rangle \in E^*$ and $y^{-1} = x/\langle y, x\rangle \in E$, where E^* is the algebraic dual space of E and $\langle \cdot, \cdot \rangle$ is the duality product between E and E^*. Both algorithms need to perform operations involving elements of the algebraic dual space E^* and E. These algorithms involve two different rules for the even lower index terms and the odd ones. The first TEA (TEA1) for computing the quantities of the first topological Shanks transformation, $\widehat{e}_k(x_n) \in E$ [4] is presented as

$$\begin{cases} \widehat{\varepsilon}_{-1}^{(n)} = 0 \in E^*, \quad n = 0, 1, \ldots, \\ \widehat{\varepsilon}_{0}^{(n)} = x_n \in E, \quad n = 0, 1, \ldots, \\ \widehat{\varepsilon}_{2k+1}^{(n)} = \widehat{\varepsilon}_{2k-1}^{(n+1)} + \dfrac{y}{\left\langle y, \widehat{\varepsilon}_{2k}^{(n+1)} - \widehat{\varepsilon}_{2k}^{(n)} \right\rangle} \in E^*, \quad n, k = 0, 1, \ldots, \\ \widehat{\varepsilon}_{2k+2}^{(n)} = \widehat{\varepsilon}_{2k}^{(n+1)} + \dfrac{\widehat{\varepsilon}_{2k}^{(n+1)} - \widehat{\varepsilon}_{2k}^{(n)}}{\left\langle \widehat{\varepsilon}_{2k+1}^{(n+1)} - \widehat{\varepsilon}_{2k+1}^{(n)}, \widehat{\varepsilon}_{2k}^{(n+1)} - \widehat{\varepsilon}_{2k}^{(n)} \right\rangle} \in E, \quad n, k = 0, 1, \ldots. \end{cases}$$

The quantities $\widetilde{e_k}(x_n) \in E$ of the second topological Shanks transformation can be recursively computed by the second TEA (TEA2) [4] as

$$\begin{cases} \widetilde{\varepsilon}_{-1}^{(n)} = 0 \in E^*, \quad n = 0, 1, \ldots, \\ \widetilde{\varepsilon}_{0}^{(n)} = x_n \in E, \quad n = 0, 1, \ldots, \\ \widetilde{\varepsilon}_{2k+1}^{(n)} = \widetilde{\varepsilon}_{2k-1}^{(n+1)} + \dfrac{y}{\left\langle y, \widetilde{\varepsilon}_{2k}^{(n+1)} - \widetilde{\varepsilon}_{2k}^{(n)} \right\rangle} \in E^*, \quad n, k = 0, 1, \ldots, \\ \widetilde{\varepsilon}_{2k+2}^{(n)} = \widetilde{\varepsilon}_{2k}^{(n+1)} + \dfrac{\widetilde{\varepsilon}_{2k}^{(n+1)} - \widetilde{\varepsilon}_{2k}^{(n)}}{\left\langle \widetilde{\varepsilon}_{2k+1}^{(n+1)} - \widetilde{\varepsilon}_{2k+1}^{(n)}, \widetilde{\varepsilon}_{2k}^{(n+2)} - \widetilde{\varepsilon}_{2k}^{(n+1)} \right\rangle} \in E, \quad n, k = 0, 1, \ldots. \end{cases}$$

Proposition 1. For $k, n = 0, 1, \ldots$, we have

$$\widehat{\varepsilon}_{2k}^{(n)} = \widehat{e_k}(x_n), \qquad \left\langle y, \widehat{\varepsilon}_{2k}^{(n)} \right\rangle = e_k(\langle y, x_n \rangle),$$
$$\widehat{\varepsilon}_{2k+1}^{(n)} = y / \langle y, \widehat{e_k}(\Delta x_n) \rangle, \quad \widehat{\varepsilon}_{2k+1}^{(n)} = y / e_k(\langle y, \Delta x_n \rangle).$$

These relations are also true for the $\widetilde{\varepsilon}_k^{(n)}$'s and $\widetilde{e}_k$'s; see [4].

Brezinski [4] investigated the convergence and acceleration results of TEA, and he and Redivo-Zaglia [8] introduced some other convergence and acceleration results that would not have been easily obtained directly from the rules of the TEA.

2.4 The STEA

In simplified versions of TEA1 and TEA2, STEA1 and STEA2, we have one rule instead of two, the functional y is utilized in the initialization of the

algorithm, the storage is reduced, and the numerical stability can be partly controlled. These new algorithms also allow for the proof of theoretical results on the convergence and acceleration of the transformation [8]. The rule of the STEA1 is

$$\widehat{\varepsilon}_{2k+2}^{(n)} = \widehat{\varepsilon}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}}(\widehat{\varepsilon}_{2k}^{(n+1)} - \widehat{\varepsilon}_{2k}^{(n)}), \quad k, n = 0, 1, \ldots, \tag{3}$$

with $\widehat{\varepsilon}_0^{(n)} = x_n, n = 0, 1, \ldots$, where $\varepsilon_k^{(n)}$'s are obtained by using Wynn's SEA to the sequence $(\langle y, x_n\rangle)$ and $\widehat{\varepsilon}_{2k}^{(n)} = \widehat{e_k}(x_n)$. Note that (3) can be written as

$$\begin{aligned}\widehat{\varepsilon}_{2k+2}^{(n)} &= \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n)}}{\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}}\widehat{\varepsilon}_{2k}^{(n+1)} - \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}}\widehat{\varepsilon}_{2k}^{(n)} \\ &= \widehat{\varepsilon}_{2k}^{(n)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n)}}{\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}}(\widehat{\varepsilon}_{2k}^{(n+1)} - \widehat{\varepsilon}_{2k}^{(n)}),\end{aligned}$$

and we have the following property:

Property 1. For all n, the computation of $\widehat{\varepsilon}_{2k+2}^{(n)}$ is stable if there exists M_k, independent of n, such that

$$\left|\frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n)}}{\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}}\right| + \left|\frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}}\right| \leq M_k. \tag{4}$$

Theorem 2. Let $r_k^{(n)} = (\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)})/(\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)})$. From (3), we have

$$\frac{\|\widehat{\varepsilon}_{2k+2}^{(n)} - \widehat{\varepsilon}_{2k}^{(n+1)}\|}{\|\widehat{\varepsilon}_{2k}^{(n+1)} - \widehat{\varepsilon}_{2k}^{(n)}\|} = |r_k^{(n)}|. \tag{5}$$

If $\lim_{n\to\infty} \widehat{\varepsilon}_{2k}^{(n)} = x$ and if there exists M such that for all $n \geq N$, $|r_k^{(n)}| \leq M$, then

$$\lim_{n\to\infty} \widehat{\varepsilon}_{2k+2}^{(n)} = x.$$

Proof. The relation (3) can be shown as

$$\widehat{\varepsilon}_{2k+2}^{(n)} = (1 + r_k^{(n)})\widehat{\varepsilon}_{2k}^{(n+1)} - r_k^{(n)}\widehat{\varepsilon}_{2k}^{(n)}. \tag{6}$$

By the assumption on $r_k^{(n)}$ and since the two scalar coefficients in the right-hand side sum up to 1, the conditions of the Toeplitz theorem for summation processes are satisfied, which proves the result [4]. □

The STEA2 with the same implementation as the first can be implemented in the following form:

$$\widetilde{\varepsilon}_{2k+2}^{(n)} = \widetilde{\varepsilon}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}}(\widetilde{\varepsilon}_{2k}^{(n+2)} - \widetilde{\varepsilon}_{2k}^{(n+1)}), \quad k, n = 0, 1, \ldots, \quad (7)$$

with $\widetilde{\varepsilon}_0^{(n)} = x_n, n = 0, 1, \ldots$ and $\varepsilon_k^{(n)}$'s are the same for the STEA1 and, we get $\widetilde{\varepsilon}_{2k}^{(n)} = \widetilde{e}_k(x_n)$.

Note that there are several possibilities for choosing the linear functional $y \in E^*$, since now, with the STEA, it appears only in the initialization terms of the SEA. When $E = \mathbb{C}^n$, we can choose it as the usual inner product

$$y : x_n \in \mathbb{C}^n \to \langle y, x_n \rangle = (y, x_n).$$

When $E = \mathbb{C}^{n\times n}$, we may set

$$y : x_n \in \mathbb{C}^{n\times n} \to \langle y, x_n \rangle = trace(x_n).$$

When $E = \mathbb{C}^{n\times m}$, we may choose a matrix $Y \in \mathbb{C}^{n\times m}$ and define

$$y : x_n \in \mathbb{C}^{n\times m} \to \langle y, x_n \rangle = trace(Y^T x_n).$$

We can also define y by $(u, x_n v)$, where $u \in \mathbb{C}^n$ and $v \in \mathbb{C}^m$ [8].

Brezinski [4] investigated the convergence and acceleration results of STEA for a sequence in vector space.

3 Numerical method based on acceleration method

In this section, to accelerate the iterative method for solving the nonlinear Volterra–Fredholm integral equation of the second kind, we suggest utilizing STEA. The first step involves converting the integral equation into a discrete form, followed by solving the system of equations using the Picard iteration

method. Lastly, to accelerate this process, we suggest employing the STEA1 and STEA2 convergence acceleration methods.

3.1 The approximation of the integral term

It is an obvious numerical method for solving integral equations to approximate their integral term using a quadrature rule. If we initially ignore the error, then the integral equation (1) will be replaced by the approximation equation. Choose a regular mesh in t, s. Let $\Delta = \{a = t_0, \ldots, t_p = b\}$ be an equidistant partition of $[a, b]$, with the partition's discretization parameter, $h = t_{i+1} - t_i,\ i = 0, \ldots, p-1$. Now if we take Δ on $[a, b]$, we have

$$x(t_i) = f(t_i) + \int_a^{t_i} g(t_i, s, x(s))ds + \int_a^b k(t_i, s, x(s))ds. \tag{8}$$

By partitioning Δ as above with $h = s_{i+1} - s_i,\ i = 0, \ldots, p-1$ and also the known weights $w_{i_j},\ j = 0, \ldots, i$ for the intervals $[a, t_i],\ i = 0, \ldots, p$ and $w_l,\ l = 0, \ldots, p$ for the intervals $[a, b]$, equality (8) can be written as

$$x(t_i) = f(t_i) + \sum_{j=0}^{i} w_{i_j} g(t_i, s_j, x(s_j)) + \sum_{l=0}^{p} w_l k(t_i, s_l, x(s_l)), \qquad i = 0, \ldots, p. \tag{9}$$

So, we set $t_i = s_i,\ i = 0, \ldots, p$ and $x_i = x(t_i),\ x_l = x(s_l),\ x_j = x(s_j),\ f_i = f(t_i),\ i = 0, \ldots, p$ for simplicity and we find approximate values $x_i,\ i = 0, \ldots, p$ as the solution of the system of $p+1$ nonlinear equations

$$x_i = f_i + \sum_{j=0}^{i} w_{i_j} g(t_i, t_j, x_j) + \sum_{l=0}^{p} w_l k(t_i, t_l, x_l), \qquad i = 0, \ldots, p. \tag{10}$$

This method is called the quadrature or Nystorm method [12].

3.2 The iterative technique

In order to solve the system of nonlinear equations (10) that implicitly define x_i, we suggest an iterative approach. Typically, this involves employing

simple iteration techniques, such as the fixed-point iteration process (Picard iteration). This approach yields a sequence of values $x_i^{(n)},\ n = 0, 1, \ldots$, which hopefully converges to a unique solution. So, applying process (10), we have

$$x_i^{(n+1)} = f_i + \sum_{j=0}^{i} w_{i_j} g(t_i, t_j, x_j^{(n)}) + \sum_{l=0}^{p} w_l k(t_i, t_l, x_l^{(n)}), \qquad i = 0, \ldots, p, \quad (11)$$

or generally, relaxation process

$$x_i^{(n+1)} = x_i^{(n)} - \alpha \Bigg\{ x_i^{(n)} - f_i - \sum_{j=0}^{i} w_{i_j} g(t_i, t_j, x_j^{(n)}) - \sum_{l=0}^{p} w_l k(t_i, t_l, x_l^{(n)}) \Bigg\}, \quad i = 0, \ldots, p, \quad n = 1, 2, \ldots, \quad (12)$$

where $x_i^{(0)}, i = 0, \ldots, p$, is the initial approximation of the solution at the points t_i, which is mostly taken as 0, 1, or t. The parameter α, which is different from 1, is used to adjust the convergence of the iterative method, and we choose its value experimentally. However, when implementing the desired extrapolation method, the convergence of the resulting sequence from the iterative method is not necessary. When $\alpha = 1$, the Picard iteration method diverges, but it may also converge at times. In various iterative methods, the value of α has been examined. For instance, in Mann's iterative method [20], the value of α changes in each iteration and is replaced by a sequence of α_n. Another method to find the appropriate value of the α parameter is dynamic relaxation [16], which is used in many other fixed-point methods [6, 18, 26, 3].

In short, we have a two-step approximation method that is supposed to yield the exact solution of x. First, we apply the Nystorm method to approximate the exact solution x at the points $t_i,\ i = 0, \ldots, p$ then obtain the approximate approximation by iterating $x^{(n)} = (x_0^{(n)}, \ldots, x_p^{(n)})$. The iterative method described above may not converge, or its convergence may be very slow. However, the vector sequence $x^{(n)}$ can be directly accelerated by an appropriate method, and we want to accelerate it by STEA1 and STEA2.

3.3 Implementation the algorithms of STEA1 and STEA2

In order to implement STEA1 and STEA2 to approximate the solution of the Volterra–Fredholm integral equation (1), we approximate the occurring integrals by the trapezoidal rule with $h = (b-a)/p$ and $t_i = a + ih$ for $i = 1, \ldots, p$, giving the following nonlinear system of equations

$$x_0 = f_0 + \frac{h}{2}\Big[k(t_0,t_0,x_0) + 2\sum_{l=1}^{p-1} k(t_0,t_l,x_l) + k(t_0,t_p,x_p)\Big],$$

$$\begin{aligned} x_i = f_i + \frac{h}{2}\Big[\big(k(t_i,t_0,x_0) + g(t_i,t_0,x_0)\big) + 2\sum_{j=1}^{i-1}\big(k(t_i,t_j,x_j) + g(t_i,t_j,x_j)\big) \\ + 2k(t_i,t_j,x_i) + g(t_i,t_j,x_i) + 2\sum_{j=i+1}^{p-1} k(t_i,t_j,x_j) + k(t_i,t_p,x_p)\Big], \\ i = 1,\ldots,p-1, \end{aligned}$$

$$\begin{aligned} x_p = f_p + \frac{h}{2}\Big[\big(k(t_p,t_0,x_0) + g(t_p,t_0,x_0)\big) + 2\sum_{j=1}^{p-1}\big(k(t_p,t_j,x_j) + g(t_p,t_j,x_j)\big) \\ + \big(k(t_p,t_p,x_p) + g(t_p,t_p,x_p)\big)\Big]. \end{aligned}$$

Hence, a system of $p+1$ nonlinear equations is created, and we want to solve it by an iterative process (12).

These iterations can be written as

$$x^{(n+1)} = F(x^{(n)}),$$

starting $x^{(0)} = 1$.

The STEA1 and the STEA2 will be applied to the sequence of vectors $x^{(n)} = (x_0^{(n)}, \ldots, x_p^{(n)})^T$ and the SEA to the sequence of scalars $(y, x^{(n)})$. We set y in two ways: it is randomly chosen in $[-1, 1]$, or it is set to $(1, \ldots, 1)^T$. The value of α is chosen experimentally to induce the convergence of the iterations [3]. To apply the algorithms, we first fix the even column and want to reach in the ε-array, say $2k$. Then, we compute $2k+1$ terms of the original sequence and, using, for example, the STEA1, we obtain the values of the

array in the order $\widehat{\varepsilon}_0^{(0)}, \widehat{\varepsilon}_0^{(1)}, \widehat{\varepsilon}_2^{(0)}, \widehat{\varepsilon}_2^{(1)}, \ldots, \widehat{\varepsilon}_{2k}^{(0)}$. The following Table 5 shows a scheme of the method.

Table 5: Implementation of STEAs

$$
\begin{cases}
\textit{Choose } 2k \textit{ and } x^0. \\
\textit{For } n = 1, 2, \ldots \\
\quad \textit{compute } x^n. \\
\quad \textit{Apply the } STEA1 \textit{ to } x^0, x^1, x^2, \ldots \textit{ and compute the sequences of} \\
\quad \textit{extrapolated values } \widehat{\varepsilon}_0^{(0)} = x^0, \widehat{\varepsilon}_0^{(1)}, \widehat{\varepsilon}_2^{(0)}, \widehat{\varepsilon}_2^{(1)}, \ldots, \widehat{\varepsilon}_{2k}^{(0)}, \widehat{\varepsilon}_{2k}^{(1)}, \widehat{\varepsilon}_{2k}^{(2)}, \ldots \\
\quad \textit{and similar quantities by the } STEA2. \\
\textit{end.}
\end{cases}
$$

4 Numerical examples

Some examples of the use of STEA1 and STEA2 to accelerate the iterations (12) for the calculation of an approximate solution of the Volterra–Fredholm integral equations of the second kind are presented in this section. The exact solutions for these examples are available. To demonstrate the accuracy and effectiveness of the proposed method, we have compared the exact solution with the approximation obtained from the Picard iterative method and the approximations obtained from the convergence acceleration methods STEA1 and STEA2. For each example, we have drawn five figures corresponding to the y vector selection. These figures include the exact solution and approximate solutions computed by using STEA1 and STEA2, the Euclidean norm of the errors, and the Euclidean norm of the differences.

Here, *sol* and x represent the exact solution and its approximation using the Picard iteration method, while *eps*1 and *eps*2 are approximate solutions computed by using STEA1 and STEA2 in accelerated Picard iterations, respectively. To stop the use of STEA1, we use the following inequalities as a condition:

$$\|\widehat{\varepsilon}_{2k}^{(n+1)} - F(\widehat{\varepsilon}_{2k}^{(n+1)})\| > L\|\widehat{\varepsilon}_{2k}^{(n)} - F(\widehat{\varepsilon}_{2k}^{(n)})\|$$

or

$$\|\widehat{\varepsilon}_{2k}^{(n)} - F(\widehat{\varepsilon}_{2k}^{(n)})\| \leq \delta.$$

These relations are also considered for STEA2, where L and δ are user-defined, and F is previously defined.

To evaluate the convergence of the proposed method to the solution of the fixed-point problem, we compare the approximate values obtained from the convergence acceleration methods STEA1 and STEA2 with the values obtained from applying the function F to them.

Example 1. Consider the following nonlinear Volterra–Fredholm integral equation with the exact solution $x(t) = t$ [27]:

$$x(t) = \frac{1}{6}t + \frac{1}{2}te^{-t^2} + \int_0^t tse^{-x^2(s)}ds + \int_0^1 tx^2(s)ds, \qquad t \in [0,1].$$

For starting, we consider $x^{(0)} = 1$ as the initial vector and we take $\alpha = 0.1$, $2k = 8$, and $p = 12$. In Figure 1, we see the exact solution and approximate solutions obtained from acceleration methods STEA1 and STEA2. Also, we see the errors and differences in Figures 2 and 3 with 50 iterations. We get the results on the left of the figures with $y = (1, \ldots, 1)^T$ and on the other hand with y random.

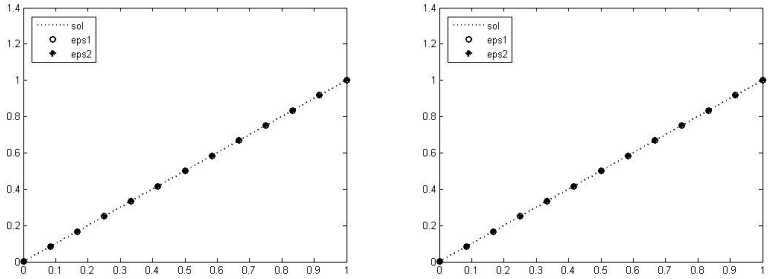
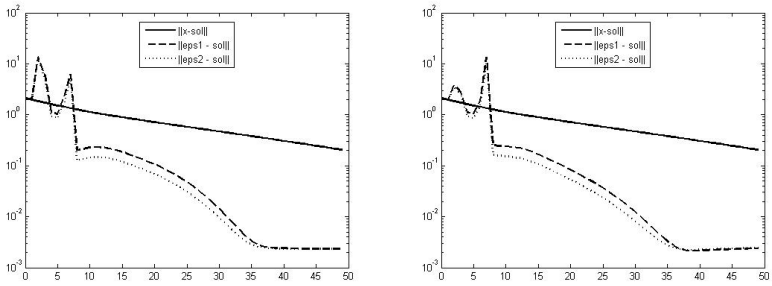
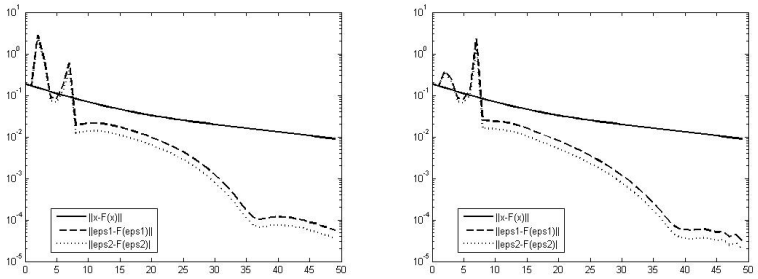


Figure 1: Exact solution and approximate solutions obtained from acceleration methods STEA1 and STEA2 for Example 1 with $y = (1, \ldots, 1)^T$ (left) and with y random (right). The exact solution is $x(t) = t$.



(a) (b)

Figure 2: Errors for Example 1 with $y = (1, \ldots, 1)^T$ (left) and with y random (right). We take $\alpha = 0.1$, $2k = 8$, and $p = 12$.



(a) (b)

Figure 3: Difference for Example 1 with $y = (1, \ldots, 1)^T$ (left) and with y random (right). We take $\alpha = 0.1$, $2k = 8$, and $p = 12$.

Table 6 shows the errors obtained by the iterative method, STEA1, and STEA2 for 50 iterations and different values of α, p, and $2k$.

According to the definition, we expected STEA2's performance to be better than STEA2's as seen in Table 6. Also, the data in Table 6 obtained by choosing random y show better results. Table 7 shows the difference between the solution obtained from the iterative method and F applied to it and those obtained from STEA1 and STEA2 and F applied to them for 50 iterations and different values of α, p, and $2k$.

Table 6: The errors between the exact solution and the solution obtained from the iterative method, as well as between the exact solution and those obtained from the acceleration methods STEA1 and STEA2. The exact solution is displayed with sol, the solution obtained from the iterative method is displayed with x, and those obtained from acceleration methods STEA1 and STEA2 are displayed, respectively, with $eps1$ and $eps2$. With $\alpha = 0.5$ and $p = 50$, the iterations stopped at 28 since $\delta = 10^{-8}$, L=20.

α	p	2k	y	$\|x - sol\|$	$\|eps1 - sol\|$	$\|eps2 - sol\|$
0.1	12	8	$(1, \ldots, 1)^T$	9.66×10^{-2}	1.10×10^{-3}	1.12×10^{-3}
0.1	12	8	random	9.66×10^{-2}	1.43×10^{-3}	1.34×10^{-3}
0.1	12	6	$(1, \ldots, 1)^T$	9.66×10^{-2}	2.21×10^{-3}	1.81×10^{-3}
0.1	12	6	random	9.66×10^{-2}	3.59×10^{-3}	2.96×10^{-3}
0.1	12	4	$(1, \ldots, 1)^T$	9.66×10^{-2}	1.90×10^{-2}	1.51×10^{-2}
0.1	12	4	random	9.66×10^{-2}	1.05×10^{-2}	8.68×10^{-3}
0.5	50	8	$(1, \ldots, 1)^T$	1.06×10^{-3}	6.87×10^{-5}	6.87×10^{-5}
0.5	50	8	random	6.45×10^{-4}	6.86×10^{-5}	6.86×10^{-5}
0.5	50	6	$(1, \ldots, 1)^T$	5.66×10^{-3}	2.85×10^{-4}	2.85×10^{-4}
0.5	50	6	random	7.33×10^{-3}	2.84×10^{-4}	2.85×10^{-4}
0.1	50	8	$(1, \ldots, 1)^T$	9.72×10^{-2}	2.72×10^{-4}	1.75×10^{-4}
0.1	50	8	random	9.72×10^{-2}	1.45×10^{-4}	9.88×10^{-5}

Example 2. Consider the following nonlinear Volterra–Fredholm integral equation with the exact solution $x(t) = e^t$ [2],

$$x(t) = e^t(1-t) + \frac{\pi}{4}t - t\tan^{-1}(e^t) + \int_0^t \frac{tx(s)}{1+x^2(s)}ds + \int_0^1 tse^t x(s)ds, \quad t \in [0,1].$$

For starting, we consider $x^{(0)} = 1$ as the initial vector, and we take $\alpha = 0.05$, $2k = 8$, and $p = 12$. In Figure 4, we see the exact solution and approximate solutions obtained from acceleration methods STEA1 and STEA2. Also, we see the errors and differences in Figures 5 and 6 with 50 iterations. We get

Table 7: The difference between the solution obtained from the iterative method and F applied to it and those obtained from STEA1 and STEA2 and F applied to them for 50 iterations and different values of α, p, and $2k$. With $\alpha = 0.5$ and $p = 50$, the iterations stopped at 28 since $\delta = 10^{-8}$, L=20.

α	p	2k	y	$\|x - F(x)\|$	$\|eps1 - F(eps1)\|$	$\|eps2 - F(eps2)\|$
0.1	12	8	$(1, \ldots, 1)^T$	8.99×10^{-3}	5.70×10^{-5}	3.66×10^{-5}
0.1	12	8	random	8.99×10^{-3}	5.86×10^{-5}	3.82×10^{-5}
0.1	12	6	$(1, \ldots, 1)^T$	8.99×10^{-3}	3.08×10^{-4}	2.23×10^{-4}
0.1	12	6	random	8.99×10^{-3}	1.99×10^{-4}	1.46×10^{-4}
0.1	12	4	$(1, \ldots, 1)^T$	8.99×10^{-3}	3.84×10^{-3}	3.08×10^{-3}
0.1	12	4	random	8.99×10^{-3}	3.11×10^{-4}	2.56×10^{-4}
0.5	50	8	$(1, \ldots, 1)^T$	9.72×10^{-4}	1.35×10^{-7}	7.77×10^{-9}
0.5	50	8	random	5.66×10^{-3}	6.15×10^{-8}	4.28×10^{-9}
0.5	50	6	$(1, \ldots, 1)^T$	1.27×10^{-3}	5.34×10^{-8}	6.36×10^{-9}
0.5	50	6	random	1.67×10^{-3}	5.29×10^{-8}	6.58×10^{-9}
0.5	50	4	$(1, \ldots, 1)^T$	5.66×10^{-4}	2.52×10^{-8}	6.12×10^{-9}
0.5	50	4	random	4.32×10^{-4}	1.08×10^{-8}	8.81×10^{-9}

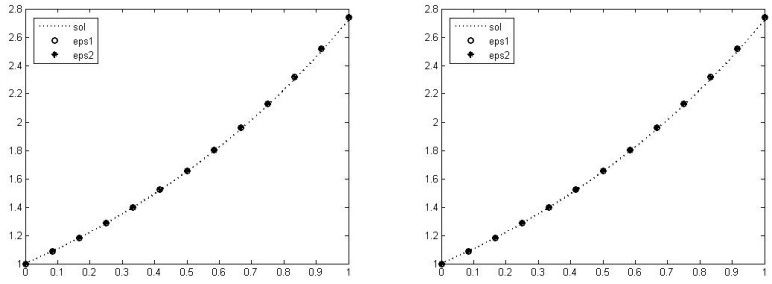
the results on the left of the figures with $y = (1, \ldots, 1)^T$ and on the other hand with y random.

Table 8 shows the errors obtained by the iterative method, STEA1, and STEA2 for 50 iterations and different values of α, p, and $2k$.

Table 9 shows the difference between the solution obtained from the iterative method and F applied to it and those obtained from STEA1 and STEA2 and F applied to them for 50 iterations and different values of α, p, and $2k$.

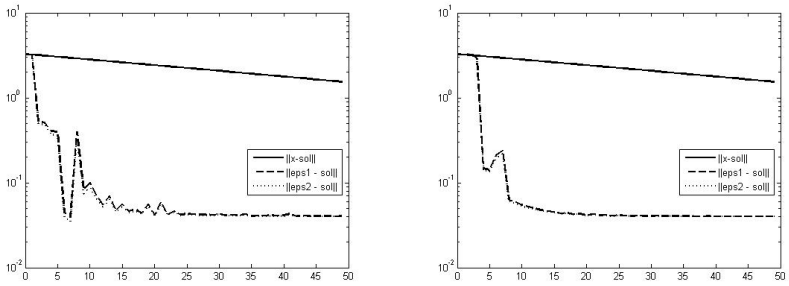
5 Conclusion

In this study, we employed simplified topological epsilon algorithms to accelerate the convergence of the relaxed Picard iteration scheme. We achieved this by discretizing the integral terms using a quadrature formula and ob-



(a) (b)

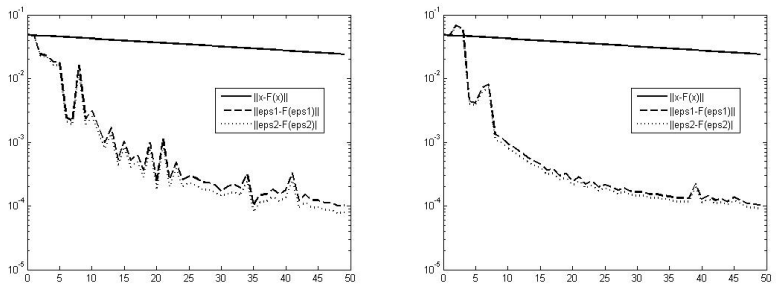
Figure 4: Exact solution and approximate solutions of STEA1 and STEA2 for Example 2 with $y = (1, \ldots, 1)^T$ (left) and with y random (right). The exact solution is $x(t) = e^t$.



(a) (b)

Figure 5: Errors for Example 2 with $y = (1, \ldots, 1)^T$ (left) and with y random (right). We take $\alpha = 0.05$, $2k = 8$, and $p = 12$.

taining a system of nonlinear equations. We also tested some examples and performed error analysis to verify the efficiency of our approach. After comparing the approximate solution obtained from the Picard iteration method with the values obtained from the convergent acceleration methods STEA1 and STEA2, it is evident that the approximate solution obtained from the convergent acceleration method is more precise and closer to the actual solution. It is suggested to use convergence acceleration methods STEA1 and STEA2 in achieving the desired approximate solution of multiple and multivariable integral equations.



(a) (b)

Figure 6: Differences for Example 2 with $y = (1, \ldots, 1)^T$ (left) and with y random (right). We take $\alpha = 0.05$, $2k = 8$, and $p = 12$.

Acknowledgements

Thank you to the referee for his careful review of the article.

References

[1] Babolian, E., Bazm, S. and Lima, P. *Numerical solution of nonlinear two-dimensional integral equations using rationalized Haar functions*, Commun. Nonlinear. Sci. Numer. Simul. 16(3) (2011), 1164–1175.

[2] Borzabadi, A.H. and Heidari, M. *A Successive numerical scheme for some classes of Volterra–Fredholm integral equations*, Iranian J. Math. Sci. Inf. 10(2) (2015), 1–10.

[3] Brezinski, C. *Numerical stability of a quadratic method for solving systems of non linear equations*, Computing 14 (1975), 205–211.

[4] Brezinski, C. *Generalisations de la transformation de shanks, de la table de Pade et de l'ε-algorithme*, Calcolo 12 (1975), 317–360.

[5] Brezinski, C. *Padé-type approximation and general orthogonal polynomials*, Birkhauser-Verlag, 1980.

Table 8: The error between the exact solution and the solution obtained from the iterative method, as well as between the exact solution and those obtained from the acceleration methods STEA1 and STEA2. The exact solution is displayed with sol, the solution obtained from the iterative method is displayed with x, and those obtained from acceleration methods STEA1 and STEA2 are displayed, respectively, with $eps1$ and $eps2$.

α	p	2k	y	$\|x - sol\|$	$\|eps1 - sol\|$	$\|eps2 - sol\|$
0.05	12	8	$(1, \ldots, 1)^T$	1.54×10^{0}	4.03×10^{-2}	4.00×10^{-2}
0.05	12	8	random	1.54×10^{0}	4.00×10^{-2}	3.99×10^{-2}
0.05	12	6	$(1, \ldots, 1)^T$	1.54×10^{0}	3.78×10^{-2}	3.74×10^{-2}
0.05	12	6	random	1.54×10^{0}	4.00×10^{-2}	3.99×10^{-2}
0.05	12	4	$(1, \ldots, 1)^T$	1.54×10^{0}	5.24×10^{-2}	5.11×10^{-2}
0.05	12	4	random	1.54×10^{0}	3.73×10^{-2}	3.73×10^{-2}
0.05	20	8	$(1, \ldots, 1)^T$	1.93×10^{0}	1.83×10^{-2}	1.80×10^{-2}
0.05	20	8	random	1.93×10^{0}	2.11×10^{-2}	2.08×10^{-2}
0.01	12	8	$(1, \ldots, 1)^T$	3.08×10^{0}	2.57×10^{-1}	2.47×10^{-1}
0.01	12	8	random	2.82×10^{0}	9.40×10^{-2}	9.15×10^{-2}
0.01	20	8	$(1, \ldots, 1)^T$	3.93×10^{0}	1.94×10^{-1}	1.87×10^{-1}
0.01	20	8	random	3.86×10^{0}	3.23×10^{-1}	3.19×10^{-1}

[6] Brezinski, C. and Chehab, J.-P. *Nonlinear hybrid procedures and fixed point iterations*, Numer. Funct. Anal. Optim. 19 (1998), 465–487.

[7] Brezinski, C. and Redivo-Zaglia, M. *Extrapolation methods: Theory and practice*, North-Holland, 1991.

[8] Brezinski, C. and Redivo-Zaglia, M. *The simplified topological ε-algorithms for accelerating sequences in a vector space*, Siam J. Sci. Comput. 36(5) (2014), A2227–A2247.

[9] Brezinski, C. and Radivo-Zagila, M. *Extrapolation methods for the numerical solution of nonlinear Fredholm integral equations*, J. Integral

Table 9: The difference between the solution obtained from the iterative method and F applied to it and those obtained from STEA1 and STEA2 and F applied to them for 50 iterations and different values of α, p, and $2k$.

α	p	2k	y	$\|x - F(x)\|$	$\|eps1 - F(eps1)\|$	$\|eps2 - F(eps2)\|$
0.05	12	8	$(1, \ldots, 1)^T$	2.40×10^{-2}	1.02×10^{-4}	7.87×10^{-5}
0.05	12	8	random	2.40×10^{-2}	1.03×10^{-4}	9.04×10^{-5}
0.05	12	6	$(1, \ldots, 1)^T$	2.40×10^{-2}	9.42×10^{-4}	8016×10^{-4}
0.05	12	6	random	2.40×10^{-2}	9.80×10^{-5}	8.90×10^{-5}
0.05	12	4	$(1, \ldots, 1)^T$	2.40×10^{-2}	7.75×10^{-4}	6.94×10^{-4}
0.05	12	4	random	2.40×10^{-2}	7.99×10^{-4}	7.26×10^{-4}
0.05	20	8	$(1, \ldots, 1)^T$	2.98×10^{-2}	9.16×10^{-5}	7.03×10^{-5}
0.05	20	8	random	2.98×10^{-2}	9.83×10^{-5}	8.74×10^{-5}
0.01	12	8	$(1, \ldots, 1)^T$	9.19×10^{-3}	2.37×10^{-3}	2.27×10^{-3}
0.01	12	8	random	8.48×10^{-3}	6.65×10^{-4}	6.64×10^{-4}
0.01	20	8	$(1, \ldots, 1)^T$	1.18×10^{-2}	1.49×10^{-3}	1.42×10^{-3}
0.01	20	8	random	1.16×10^{-2}	9.32×10^{-4}	9.18×10^{-4}

Equ. Appl. 31 (2019), 29–57.

[10] Brezinski, C. and Sadok, H. *Lanczos-type algorithms for solving systems of linear equations*, Appl. Numer. Math. 11 (1993), 443–473.

[11] Brunner, H. *On the numerical solution of nonlinear Volterra–Fredholm integral equation by collocation methods*, SIAM J. Numer. Anal. 27(4) (1990), 987–1000.

[12] Delves, L.M. and Mohamed, J.M. *Computational methods for integral equations*, Cambridge University Press, 1985.

[13] Ghasemi, M., Tavassoli Kajani, M. and Babolian, E. *Numerical solutions of the nonlinear Volterra–Fredholm integral equations by using Homotopy Perturbation Method*, Appl. Math. Comput. 188(1) (2007), 446–449.

[14] Jbilou, K., Messaoudi, A. and Tabaa, K. *On some matrix extrapolation methods*, C. R. Math. Acad. Sci. Paris 341 (2005), 781–786.

[15] Jbilou, K. and Sadok, H. *Vector extrapolation methods. Applications and numerical comparison*, J. Comput. Appl. Math. 122 (2000), 149–165.

[16] Kuttler, U. and Wall, W.A. *Fixed-point fluid-structure interaction solvers with dynamic relaxation*, Comput. Mech. 43 (2008), 61–72.

[17] Laeli Dastjerdi, H. and Nili Ahmadabadi, M. *The numerical solution of nonlinear two-dimensional Volterra–Fredholm integral equations of the second kind based on the radial basis functions approximation with error analysis*, Appl. Math. Comput. 293 (2017), 545–554.

[18] MacLeod, A.J. *Acceleration of vector sequences by multi-dimensional* Δ^2 *methods*, Commun. Appl. Numer. Meth. 2 (1986), 385–392.

[19] Maleknejad, K., Almasieh, H. and Roodaki, M. *Triangular functions (TF) method for the solution of nonlinear Volterra–Fredholm integral equations*, Commun. Nolin. Sci. Numer. Simul. 15(11) (2010), 3293–3298.

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

[21] Mirzaee, F. and Hadadiyan, E. *Numerical solution of Volterra–Fredholm integral equations via modification of hat functions*, Appl. Math. Comput. 280 (2016), 110–123.

[22] Mosa, G.A., Abdou, M.A. and Rahby, A.S. *Numerical solutions for nonlinear Volterra–Fredholm integral equations of the second kind with a phase lag*, AIMS Math. 6 (2021), 8525–8543.

[23] Ordokhani, Y. *Solution of nonlinear Volterra–Fredholm–Hammerstein integral equations via rationalized Haar functions*, Appl. Math. Comput. 180(2) (2006), 436–443.

[24] Patchpatte, B.G. *On a class of Volterra and Fredholm non-linear integral equations*, Sarajevo J. Math. 16 (2008), 61–71.

[25] Salam A. and Graves-Morris, P.R. *On the vector ε-algorithm for solving linear systems of equations*, Numer. Algorithms 29 (2002), 229–247.

[26] Sedogbo, G.A. *Some convergence acceleration processes for a class of vector sequences*, Appl. Math. (Warsaw) 24 (1997), 299–306

[27] Shali, J.A. and Ebadi, G. *Approximate solutions of nonlinear Volterra–Fredholm integral equations*, Int. J. Non. Sci. 14(4) (2012), 425–433.

[28] Shanks, D. *Nonlinear transformations of divergent and slowly convergent sequences*, J. Math. Phys. 34 (1955), 1–42.

[29] Sidi, A. *Practical extrapolation methods: Theory and applications*, Cambridge University Press, 2003.

[30] Wazwaz, A.M. *Linear and nonlinear integral equations: Methods and applications,* Springer-Verlag, 2011.

[31] Wynn, P. *On a device for computing the $e_m(S_n)$ transformation*, Math. Tables Aids Comput. 10 (1956), 91–96.

[32] Wynn, P. *Acceleration techniques for iterated vector and matrix problems*, Math. Comp. 16 (1962), 301–322.

[33] Yalcinbas, S. *Taylor polynomial solutions of nonlinear Volterra–Fredholm integral equations*, Appl. Math. Comput. 127 (2002), 195–206.

[34] Yousefi, S. and Razzaghi, M. *Legendre wavelets method for the nonlinear Volterra–Fredholm integral equations*, Math. Comput. Simul. 70(1) (2005), 1–8.