Università degli studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea Magistrale in Matematica

Extrapolation methods for the numerical solution of nonlinear Urysohn integral equations

A study on univariate and bivariate domain

Relatrice:
Prof. Michela Redivo Zaglia

Laureando:
Riccardo Viero, 1176556

13 Dicembre 2019

Anno Accademico 2018-2019

Contents

Introduction **5**

1 Iterative method for integral equations **7**
1.1 The existence and uniqueness of the solution 8
1.2 The Nyström Method 13
1.3 Picard iteration to solve nonlinear systems 18
1.4 Numerical results 22

2 Extrapolation Methods for integral equations **27**
2.1 Shanks' transformation for scalar sequence 28
2.2 ε-algorithm and Cordellier's particular rule 31
2.3 The simplified topological ε-algorithm 35
2.4 Numerical results with fixed step size 38
2.5 A new adaptive strategy for step size 47

3 Comparing the fixed and the adaptive step size strategies **57**
3.1 Discussion about the parameters 58
3.2 Examples in bivariate domain 67
3.2.1 Examples in square 67
3.2.2 Examples in disk 73

Conclusion **77**

A Reference tables for fixed step size strategy **79**

B Reference tables for adaptive step size strategy **89**

C MATLAB software **99**

Bibliography **111**

Integral Equations notations

Ω	=	The domain where Integral Equation is defined
u	=	A continuous function defined in Ω
$(u^{(n)})$	=	A sequence of continuous function defined in Ω
u^*	=	The continuous solution of the Integral Equation
$\mathcal{T}$	=	The contraction operator defined on continuous functions
L	=	The Lipschitz's constant of integral function K
$[-\zeta, \zeta]$	=	The interval where the Lipschitz condition of the kernel of integral function K is verified
$\mathcal{E}^{\mathrm{QUAD}}(u)$	=	The error associated to quadrature formula where we use a function u like a solution
(x_i, w_i)	=	The $p+1$ nodes and the associated positive weights of quadrature formula
$\mathbf{T}$	=	The operator given by $\mathcal{T}$ according to the Nyström Method defined on vector space $\mathbb{R}^{p+1}$
$\hat{u}$	=	The optimal approximated solution given by the Nyström Method
$\mathbf{u}$	=	A vector of dimension $p+1$
$\tilde{u}$	=	The approximated solution given by the Nyström Method associated to a general vector $\mathbf{u}$ (useful when we estimate the error, etc.)
$\bar{\mathbf{u}}$	=	The vector solution of the nonlinear system given by the Nyström Method
$\mathbf{u}^*$	=	The evaluations of function u^* in the quadrature nodes
$\bar{\mathbf{u}}$	=	The vector solution of the nonlinear system given by the Nyström Method
$\mathbf{v} \equiv c$	$\Rightarrow$	The vector $\mathbf{v}$ having al components equal to the constant c

Extrapolation notations

(S_n)	=	The scalar original sequence
S	=	The limit of scalar original sequence (S_n)
T	=	The sequence transformation
(T_n)	=	The scalar extrapolated sequence
$e_k(S_n)$	=	The Shanks transformation of order k
$\varepsilon_k^{(n)}$	=	A scalar element of ε-array of Wynn's algorithm
$(\mathbf{S}_n)$	=	The vector original sequence
$\mathbf{S}_n$	=	The limit of vector sequence $(\mathbf{S}_n)$
$\mathbf{y}$	=	The dual vector associated to the STEA
$\tilde{\mathbf{e}}_k(\mathbf{S}_n)$	=	The second simplified topological Shanks transformation STEA2
$\widetilde{\boldsymbol{\varepsilon}}_k^{(n)}$	=	A vector element of STEA2 algorithm

Introduction

In their recent article [10] Brezinski and Redivo-Zaglia propose a new competitive approach to solve a special subclass of nonlinear Fredholm integral equations of the second kind, called *Urysohn integral equations* in one variable. In their paper they focus on the exemplification of the performance of simplified topological ε-algorithm [11] applied to the underlying system of nonlinear equations generated by the Nyström Method in order to accelerate the convergence. In this thesis we want to present the results of my research that has allowed us to understand the underlying theoretical and empirical aspects. The performances obtained through our considerations are better than those originally published. A novelty that we propose in this thesis is the extension of this technique to the study of Urysohn integral equation defined on bivariate domain.

The aim of the first chapter is to deepen the theoretical aspect of Brezinski and Redivo-Zaglia's approach: in particular, we formulate some conditions under which the error is strictly related to the residual and to the choice of quadrature formula used in Nyström Method. We present the Relaxed Picard Iteration Method that generates a sequence of vectors that converges to the solution of underlying nonlinear system.

In the second chapter we present the simplified topological ε-algorithm and we apply it to the sequence generated by the fixed step size strategy, as done by the authors of [10]. In order to try to increase the performance of the extrapolation we propose an adaptive strategy, which is more complex than the fixed step size strategy, and generates a sequence close to the kernel of extrapolation. In this way we hoped to achieve a better performance respect to the results obtained using the fixed step size strategy.

In the third chapter the comparison between the two strategies, each of one with the best parameters, has shown that often they coincide or the fixed one is slightly better. These experiments indicate a preference in the choice of parameters if the above mentioned conditions are satisfied. Thanks to the results we have reached we decided to compare different numerical recent works that study a nonlinear integral equation defined on multivariate domain with our method,

which has proven to be more precise and computationally competitive.

In the Appendices A and B we show the performances obtained using one of the two strategies varying the parameters for every examples we have presented in the last chapter. While in the Appendix C it is shown the MATLAB software I have implemented in order to establish the results of this thesis.

Chapter 1

Iterative method for integral equations

Generally speaking, an integral equation is an equation for an unknown function u, where u appears also under the integral sign. The integral equations are classified (see [18, §1]) by various properties and we present some of the most common in the following. If integral domain is a fixed subset of the $\mathbb{R}^d$ we have a *Fredholm integral equation*, otherwise if integral domain varies with the variable we have a *Volterra integral equation*. Another classification distinguishes *Integral equations of the first kind* from *Integral equations of the second kind*: in the former the unknown function appears only under the integral sign, in the latter the unknown function appears also outside the integral. Finally we distinguish *linear integral equations* from *nonlinear integral equations* depending on whether the equation is linear with respect to the unknown function or not.

In our work we study the **Urysohn integral equations**, a special subclass of nonlinear Fredholm integral equations of the second kind. Its form is:

$$u(t) = f(t) + \int_\Omega K(t, x, u(x))\,dx, \quad \text{for } t \in \Omega \tag{1.1}$$

where $\Omega \subset \mathbb{R}^d$ is a compact set of a finite-dimensional space of non-zero Lebesgue measure, $K : \Omega \times \Omega \times \mathbb{R} \to \mathbb{R}$ and $f : \Omega \to \mathbb{R}$ are given functions, and $u : \Omega \to \mathbb{R}$ is an unknown function.
The last term of (1.1) can be interpreted as a nonlinear operator which acts in some Banach space E[1] of measurable functions on Ω,

$$\mathcal{K}[u] := \int_\Omega K(\cdot, x, u(x))\,dx \tag{1.2}$$

[1]When we study the solutions in E, we lose those ones which do not belong to E (if such solutions exist). If the operator equation, which we will see in (1.3), can be examined in different spaces, then the choice of the space E depends on the type of solutions we are interested in.

Furthermore if $f \in E$ then the Urysohn equations (1.1) are equivalent to nonlinear operator equations in E:

$$u = f + \mathcal{K}[u] \tag{1.3}$$

1.1 The existence and uniqueness of the solution

The solutions of (1.3) are the fixed points of the operator $\mathcal{T}[u] := f + \mathcal{K}[u]$. Fixed point theorems are the main tools used to prove the existence of solutions of (1.3) and hence of (1.1). The most widely used tools in the analysis of fixed points are the *Banach Contraction Mapping principle* and the *Schauder principle*. In this section we formulate an existence and uniqueness theorem for the solutions (1.1) based on Banach Contraction Mapping principle (BCM principle).

We introduce the following definition to present the BCM principle.

Definition 1.1. *An operator $A[\cdot]$ acting in a Banach space E is said to satisfy a **Lipshitz condition** with constant $\widetilde{L}$ on a set $M \subset E$, if $\forall u, v \in M$*

$$\|A[u] - A[v]\| \leq \widetilde{L}\ \|u - v\|$$

where $\|\cdot\|$ is a complete norm of the Banach space E.
*If $\widetilde{L} < 1$, then A is called **contraction operator**.*

Theorem 1.1 (Banach Contraction Mapping principle)**.** *Given a Banach space E, if the contraction operator A maps a closed set $M \subset E$ into itself, then it has a unique fixed-point u^* in M. This fixed point can be obtained as the limit of the sequence*

$$u^{(n+1)} = A[u^{(n)}]$$

where $u^{(0)}$ is an arbitrary element of M.

Proof. The proof is constructive and it's inspired by Theorem 3.1 of [13]. Let $u^{(0)}$ be an arbitrary element in M. First of all we show that $u^{(n)}{}_{n\in\mathbb{N}}$ is a Cauchy sequence. From the definition of contraction operator and the triangle inequality, if $n \geq m \geq 0$ we have:

$$\begin{aligned}
\left\|u^{(n)} - u^{(m)}\right\| &= \left\|A^n[u^{(0)}] - A^m[u^{(0)}]\right\| \leq \widetilde{L}^m \left\|A^{n-m}[u^{(0)}] - u^{(0)}\right\| \\
&\leq \widetilde{L}^m \left\|\left(A^{n-m}[u^{(0)}] - A^{n-m-1}[u^{(0)}]\right) + \ldots + \left(A^1[u^{(0)}] - u^{(0)}\right)\right\| \\
&\leq \widetilde{L}^m \sum_{i=0}^{n-m-1} \left\|A^{n-m-i}[u^{(0)}] - A^{n-m-i-1}[u^{(0)}]\right\| \\
&\leq \widetilde{L}^m \sum_{i=0}^{n-m-1} \widetilde{L}^i \left\|A[u^{(0)}] - [u^{(0)}]\right\| \leq \widetilde{L}^m \sum_{i=0}^{+\infty} \widetilde{L}^i \left\|A[u^{(0)}] - [u^{(0)}]\right\| \\
&= \frac{\widetilde{L}^m}{1 - \widetilde{L}} \left\|A[u^{(0)}] - [u^{(0)}]\right\|
\end{aligned}$$

Given an arbitrary $\epsilon > 0$, we have a large $N \in \mathbb{N}$ so that $\widetilde{L}^N < \epsilon \frac{1-\widetilde{L}}{\left\|A[u^{(0)}]-u^{(0)}\right\|}$. Therefore, by choosing m and n greater than N we have $\left\|u^{(n)} - u^{(m)}\right\| \leq \epsilon$, which implies that $u^{(n)}{}_{n\in\mathbb{N}}$ is a Cauchy sequence. This sequence converges to an element $u^* \in E$ because E is a Banach space. Moreover, $u^* \in M$ because every element of the sequence stays in the same closed set, $(u^{(n)})_{n\in\mathbb{N}} \subset M$. □

Now we want to informally discuss sufficient conditions in order to have $\mathcal{T}$ as a contraction operator. First of all it's necessary to define the Banach space in which we operate. From a practical point of view we prefer to choose the space of continuous function on Ω, $E = C(\Omega)$ with the natural norm $\|\cdot\|_\infty$. We observe that:

$$\begin{aligned}\|\mathcal{T}[u] - \mathcal{T}[v]\|_\infty &= \|f + \mathcal{K}[u] - f - \mathcal{K}[v]\|_\infty \\ &= \|\mathcal{K}[u] - \mathcal{K}[v]\|_\infty \\ &= \left\| \int_\Omega (K(\cdot, x, u(x)) - K(\cdot, x, v(x)))\, dx \right\|_\infty\end{aligned}$$

The fact that the kernel of integral equation satisfies a Lipschitz condition for the third variable, i.e. if exist L such that $L|\Omega| < 1$ and for every $u, v \in \mathbb{R}$ and $t, x \in \Omega$

$$|K(t, x, u) - K(t, x, v)| \leq L\,|u - v| \tag{1.4}$$

is sufficient to have $\mathcal{T}[\cdot]$ as a contraction operator[2]. The term $|\Omega|$ denotes the measure of the set Ω.

However it's unlikely that the kernel of integral equations globally satisfies the Lipschitz condition, therefore we assume[3] that Lipschitz condition is verified $\forall u, v, \in [-\zeta, \zeta]$. This assumption reduce the space where we search the solutions to the continuous functions with image on $[-\zeta, \zeta]$ which is denoted by $M := C(\Omega; [-\zeta, \zeta])$. In order to apply the BCM principle we must be sure that the restriction of $\mathcal{T}$ on the subset M acts on M, $\mathcal{T}[M] \subset M$: it is the most difficult part when using this principle in practice. Assuming

$$\max_{t\in\Omega,\, u\in C(\Omega;[-\zeta,\zeta])} \left| f(t) + \int_\Omega K(t, x, u(x))\, dx \right| \leq \zeta \tag{1.5}$$

we have $\forall \tilde{u} \in C(\Omega; [-\zeta, \zeta]), \mathcal{T}[\tilde{u}] \in C(\Omega; [-\zeta, \zeta])$ because $\forall \tilde{t} \in \Omega$,

$$|\mathcal{T}[\tilde{u}](\tilde{t})| \leq \max_{t\in\Omega,\, u\in C(\Omega;[-\zeta,\zeta])} \left| f(t) + \int_\Omega K(t, x, u(x))\, dx \right| \leq \zeta$$

In the next page we sum up our considerations in a theorem stating the existence and uniqueness of the solution of (1.1) (we take a cue from monographs [32, p. 390] and [19, p. 269]).

[2] We see in the next section that the property (1.4) is important for our numerical treatment and no only to prove that $\mathcal{T}[\cdot]$ is a contractor operator.

[3] We can consider a more general set where we verify the Lipschitz condition but we have more difficulties to apply the BCM principle.

Theorem 1.2. *Let $\Omega \subset \mathbb{R}^d$ be a measurable and closed set. We consider the Urysohn equation in the Banach space $E = C(\Omega)$,*

$$u(t) = f(t) + \int_\Omega K(t, x, u(x))\, dx, \quad \text{for } t \in \Omega.$$

If exist positive $\zeta, L \in \mathbb{R}$ such that

i) f and K are continuous in Ω and in $\Omega \times \Omega \times \mathbb{R}$, respectively

ii) $|K(t, x, u) - K(t, x, v)| \leq L|u - v| \quad \forall t, x \in \Omega,\ \forall |u| \leq \zeta, \forall |v| \leq \zeta$,

iii) $L \cdot |\Omega| < 1$,

iv) $\max_{t\in\Omega,\, u\in C(\Omega;[-\zeta,\zeta])} \left| f(t) + \int_\Omega K(t, x, u(x))\, dx \right| \leq \zeta$,

then the Urysohn equation has a unique continuous function, u^*, satisfying the inequality $|u^*(x)| \leq \zeta\ \forall x \in \Omega$. If $u^{(0)}$ is an arbitrary continuous function that satisfies the inequality $|u^{(0)}(x)| \leq \zeta,\ \forall x \in \Omega$ then the sequence

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

converge to u^* uniformly on Ω.

Now we want to show an example where our hypotheses hold and we obtain a solution in $M = C(\Omega; [-\zeta, \zeta]) \subset E$ meanwhile in the Banach space $E = C(\Omega)$ there is another solution, according to the note in page 3.

Observation 1.1. *We consider the third example formulated by Anderson in his article [3, p. 558] and analysed in Brezinski, Redivo-Zaglia article's [10, p. 19]. It studies the following Urysohn integral equation:*

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

This problem describes an operator that satisfies the conditions of previous theorem. In fact i) is verified because f and K are continuous in Ω and in $\Omega \times \Omega \times \mathbb{R}$, respectively.

$$L = \max_{t,x\in[a,b]\, |u|\leq\zeta} \left| \frac{\partial}{\partial u} K(t, x, u) \right| = \frac{3\sqrt{2}\pi}{8} \max_{|u|\leq\zeta} |u| = \frac{3\sqrt{2}\pi}{8}\zeta$$

We have to choose $\zeta \in (0, \frac{4}{3\sqrt{2}\pi})$ so that the iii) property is verified, i.e. $L\,|\Omega| < 1$.

The last condition iv) is verified if $\zeta \in \left(\frac{1}{4}, \frac{4}{3\sqrt{2}\pi}\right)$. *In fact we observe that:*

$$\max_{t\in\Omega,\, u\in C(\Omega;[-\zeta,\zeta])} \left| f(t) + \int_\Omega K(t,x,u(x))\,dx \right| =$$

$$= \max_{t\in\Omega, u\in C(\Omega;[-\zeta,\zeta])} \left| -\frac{1}{4}\cos\left(\frac{\pi t}{4}\right) + \frac{3\sqrt{2}\pi}{16}\int_{-1}^{1} \underbrace{\cos\left(\frac{\pi(t-x)}{4}\right)}_{\geq 0 \forall t,x\in\Omega} u^2(x)\,dx \right|$$

hence the maximizer is a constant function: $u \equiv 0$ *or* $u \equiv \zeta$

$$= \max_{t\in\Omega, l\in\{0,\zeta\}} \left| -\frac{1}{4}\cos\left(\frac{\pi t}{4}\right) + \frac{3\sqrt{2}\pi}{16} l^2 \int_{-1}^{1} \cos\left(\frac{\pi(t-x)}{4}\right) dx \right|$$

$$= \max_{t\in\Omega, l\in\{0,\zeta\}} \left| -\frac{1}{4}\cos\left(\frac{\pi t}{4}\right) + \frac{3\sqrt{2}\pi}{16} l^2 \frac{4\sqrt{2}}{\pi}\cos\left(\frac{\pi t}{4}\right) \right|$$

$$= \max_{t\in\Omega} \left\{ \frac{1}{4}, \left| \left(-\frac{1}{4} + \frac{3\zeta^2}{2}\right)\cos\left(\frac{\pi t}{4}\right) \right| \right\}$$

$$= \max_{t\in\Omega} \left\{ \frac{1}{4}, \left| -\frac{1}{4} + \frac{3\zeta^2}{2} \right| \right\}$$

Therefore for having $\max_{t\in\Omega,|u|\leq\zeta} \left| f(t) + \int_\Omega K(t,x,u(x))dx \right| \leq \zeta$, ζ *has to satisfy the following system:*

$$\begin{cases} 0 < \zeta < \frac{4}{3\sqrt{2}\pi} \\ \frac{1}{4} \leq \zeta \\ -\frac{1}{4} + \frac{3\zeta^2}{2} \leq \zeta \end{cases} \quad i.e. \quad \begin{cases} \frac{1}{4} \leq \zeta < \frac{4}{3\sqrt{2}\pi} \\ 6\zeta^2 - 4\zeta - 1 \leq 0 \end{cases} \quad i.e. \quad \begin{cases} \frac{1}{4} \leq \zeta < \frac{4}{3\sqrt{2}\pi} \\ \frac{1}{6}\left(2 - \sqrt{10}\right) \leq \zeta \leq \frac{1}{6}\left(2 + \sqrt{10}\right) \end{cases}$$

i.e. $\zeta \in \left(\frac{1}{4}, \frac{4}{3\sqrt{2}\pi}\right)$. *[25] proposes an analytic treatment and establishes the form of solutions in many specific types of integral equations. For the current example we deduce from [25, 8.3.8 p.465 and 8.8.19 p.483] that the solutions have the form:*

$$u^*(t) = \lambda_1 \cos\left(\frac{\pi t}{4}\right) + \lambda_2 \sin\left(\frac{\pi t}{4}\right) + f(t)$$

and by substitution we derive that we have two solutions

$$u_1^*(t) = \cos\left(\frac{\pi t}{4}\right) \qquad u_2^*(t) = -\frac{1}{5}\cos\left(\frac{\pi t}{4}\right)$$

where only the second belong in $C(\Omega;[-\zeta,\zeta])$ *with* $\zeta \in \left(\frac{1}{4}, \frac{4}{4\sqrt{2}\pi}\right)$.

As we will analyse later in the Theorem 1.3, the numerical treatment presented in our work acts in the confidential region defined by BCM principle hence we reach the solution u_2^*. We have already observed that the algorithm in Brezinski and Redivo-Zaglia's article [10] acts outside this confidential region and considers u_1^* as reference solution. In the Chapter 2 when we will present the extrapolation method that they have used, we will show that their results deeply depend from the starting vector according to the absence of sufficient conditions. In particular we will show that with some starting vectors the algorithm converges to u_2^*, while with some other starting vectors it turns into the confidential region (hence it converges to u_1^*), and finally in some specific cases it doesn't converge to any solution.

Before ending this section we want to discuss the last condition of the Theorem 1.2. In the Brezinski, Redivo-Zaglia's article [10] this condition is not mentioned and we present a simple example that shows that this hypothesis is important both for theoretical and practical point of view.

Observation 1.2. *We consider the previous example with different function $f(\cdot)$*

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

The properties $i)\, ii)\, iii)$ are always verified for $\zeta \in \left(0, \frac{4}{3\sqrt{2}\pi}\right)$ as in the previous example. Furthermore this is necessary to guaranty the Lipschitz property of kernel function. By repeating the previous algebraic step the last condition $iv)$ is equivalent to

$$\max\left\{\frac{1}{4}, 1 + \frac{3\zeta^2}{2}\right\} < \zeta$$

consequently $\zeta \in \left(\frac{1}{4}, \frac{4}{3\sqrt{2}\pi}\right)$ and $3\zeta^2 - \zeta + 2 < 0$ but this paraboloid is always positive. Therefore the last condition of the Theorem 1.2 isn't satisfied by any ζ.

We are aware that the Theorem 1.2 expresses only sufficient condition for existence but in this case we show that the last and how not assuming the seemingly less significant condition, there is an example of a integral equation without solution. In fact if we suppose that a solution, u^*, exists and that satisfies the inequality $|u^*(t)| \leq \frac{4}{3\sqrt{2}\pi}\ \forall t \in [-1, 1]$, in particular for $t = 0$ we must have

$$\left|\frac{3\sqrt{2}\pi}{16} \int_{-1}^{1} \cos\left(\frac{\pi\, x}{4}\right) u^2(x)\, dx + 1\right| \leq \frac{4}{3\sqrt{2}\pi}$$

But since $\cos\left(\frac{\pi\, x}{4}\right) > 0\, \forall x \in [-1, 1]$ and $1 > \frac{4}{3\sqrt{2}\pi}$ the previous inequality is never true, and this is an absurd.

Being aware of these cases is important since a numerical method could also provide a meaningless discrete solution.

1.2 The Nyström Method

An obvious idea to solve the Urysohn integral equation (1.1) is to approximate the integral with a quadrature formula. This approach offers a simpler situation in Urysohn integral equation than Volterra equation because the integration domain is fixed. Let $\{x_0, \ldots, x_p\} \subset \Omega$ be the nodes of quadrature formula with associated positive weights $\{w_0, \ldots, w_p\}$, the Urysohn integral equation (1.1) is approximated to

$$\hat{u}(t) := f(t) + \sum_{i=0}^{p} w_i \, K(t, x_i, \bar{u}_i) \quad \text{for } t \in \Omega \tag{1.6}$$

where $\bar{\mathbf{u}} \in \mathbb{R}^{p+1}$, $\bar{\mathbf{u}} = (\bar{u}_0, \ldots, \bar{u}_p)$, is the solution of nonlinear system[4]

$$u_k = f(x_k) + \sum_{i=0}^{p} w_i \, K(x_k, x_i, u_i) \quad \text{for } k = 0, \ldots, p$$

In order to shorten the notation, we introduce the vector $\mathbf{f} = (f_0, \ldots, f_p)$ given by $f_k := f(x_k)$ and the family of continuous real functions $(\mathbf{K})_{i,k=0,\ldots,p}$ given by $\mathbf{K}_{k,i}(u) := K(x_k, x_i, u)$. Thanks of this notation we can rewrite the precedent nonlinear system as

$$u_k = f_k + \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i) \quad \text{for } k = 0, \ldots, p \tag{1.7}$$

In the sequel, we want to describe an estimation of the error due to the approximation (1.6). For a function $u \in C(\Omega)$ the quadrature formula integrates the function $\varphi(x) := K(t, x, u(x))$, varying $t \in \Omega$. In order to quantify the error that results from this approximation, we introduce the quantity:

$$\mathcal{E}^{\mathrm{QUAD}}(u) := \max_{t \in \Omega} \left| \int_{\Omega} K(t, x, u(x)) \, dx - \sum_{i=0}^{p} w_p \, K(t, x_i, u(x_i)) \right| \tag{1.8}$$

We will see that the quantity $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ plays a rule in every following inequalities and it determines the validity of our numerical results as we shall see in (1.16).

Observation 1.3. *The term $\mathcal{E}^{QUAD}(u^*)$ depends on the quadrature rule chosen. For particular rules this quantity can be estimated and a wide discussion would be necessary in order to decide the number of nodes needed. In the literature of quadrature formulas it is usual to know an appropriate estimation of the $\left\| \frac{\partial^n}{\partial x} \varphi \right\|_\infty$ but our attempts to obtain it have been vain.*

[4]We will discuss about the existence and uniqueness of the solution in the section §1.3. We note that in the original Nyström Method, focused on Fredholm linear integral equations, this system is linear therefore it's simpler than our case (see [18, §4] for more details).

Now we present a simple observation that we will use in the following propositions. We remember that $u^* \in C(\Omega; [-\zeta, \zeta])$ denotes the solution of Urysohn integral equation while $\mathbf{u}^* \in \mathbb{R}^{p+1}$ represents the evaluations of u^* on the quadrature nodes. Now we suppose that the quadrature formula is at least of the order one, i.e. $\sum_{i=0}^{p} w_i = |\Omega|$. This hypothesis is verified if we consider an interpolation quadrature formula. We do not intend to deepen the construction and the property of a quadrature formula: for our use it is crucial that it well approximates the function φ and selects a finite number of points of Ω in the lowest possible number.

Lemma 1.1. *We assume the hypotheses $i) - iv)$ of the Theorem 1.2.*
Then we have the following estimation of the error between the exact solution $\bar{\mathbf{u}}$ of nonlinear system (1.7) and the vector $\mathbf{u}^*$ with respect to the $\|\cdot\|_\infty$ norm of the vector space $\mathbb{R}^{p+1}$:

$$\|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty \leq \frac{1}{1 - L\,|\Omega|} \mathcal{E}^{QUAD}(u^*) \tag{1.9}$$

Proof. In order to prove (1.9) we concatenate the following inequalities:

$$\begin{aligned}
\|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty &= \max_{k=0,\dots,p} \left| \cancel{f_k} + \sum_{i=0}^{p} w_i\, K(x_k, x_i, \bar{u}_i) - \cancel{f_k} - \int_\Omega K(x_k, x, u^*(x))\, dx \right| \\
&\qquad \textit{adding and subtracting } \sum_{i=0}^{p} w_i\, K(t, x_i, u^*(x_i)) \\
&\leq \mathcal{E}^{\mathrm{QUAD}}(u^*) + \max_{k=0,\dots,p} \left| \sum_{i=0}^{p} w_i \Big(\mathbf{K}_{k,i}(\bar{u}_i) - \mathbf{K}_{k,i}(u_i^*) \Big) \right| \\
&\leq \mathcal{E}^{\mathrm{QUAD}}(u^*) + \max_{k=0,\dots,p} \left\{ \sum_{i=0}^{p} w_i \Big| \mathbf{K}_{k,i}(\bar{u}_i) - \mathbf{K}_{k,i}(u_i^*) \Big| \right\} \\
&\leq \mathcal{E}^{\mathrm{QUAD}}(u^*) + L \sum_{i=0}^{p} w_i\, |\bar{u}_i - u_i^*| \\
&\leq \mathcal{E}^{\mathrm{QUAD}}(u^*) + L \underbrace{\left(\sum_{i=0}^{p} w_i \right)}_{|\Omega|} \|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty
\end{aligned}$$

Finally we carry $L\,|\Omega|\,\|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty$ to the left in order to obtain

$$(1 - L\,|\Omega|)\ \|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty \leq \mathcal{E}^{\mathrm{QUAD}}(u^*)$$

Since $L\,|\Omega| < 1$, we can divide for $(1 - L\,|\Omega|) > 0$ to obtain (1.9). □

We denote with $\tilde{u}(\cdot)$ the approximation given by (1.6) in which we use $\mathbf{u} = (u_0, \ldots, u_p)$ instead of $\bar{\mathbf{u}} = (\bar{u}_0, \ldots, \bar{u}_p)$,

$$\tilde{u}(t) := f(t) + \sum_{i=0}^{p} w_i \, K(t, x_i, u_i) \quad \text{for } t \in \Omega \tag{1.10}$$

where $\mathbf{u}$ stay in $\widetilde{M} := [-\zeta, \zeta]^{p+1}$ to be able to apply the Lipshitz relation (1.4). We introduce now $\tilde{u}(t)$ because numerical methods not necessarily determine the exact solution $\bar{\mathbf{u}}$ of nonlinear system (1.7) and we want determine a more inclusive error estimation.

Proposition 1.1. *We assume that the hypotheses $i) - iv)$ of the Theorem 1.2 hold. If $\mathbf{u} \in \widetilde{M}$ and $\tilde{u} \in C(\Omega)$ given by (1.10), we have the following estimation of the error between continuous function $\tilde{u}$ and u^* with respect to the natural norm of $C(\Omega)$ functions space:*

$$\|\tilde{u} - u^*\|_\infty \leq \mathcal{E}^{QUAD}(u^*) + L\,|\Omega| \; \|\mathbf{u} - \mathbf{u}^*\|_\infty \tag{1.11}$$

Furthermore if $\mathbf{u} \equiv \bar{\mathbf{u}}$ then $\tilde{u} = \hat{u}$ and the relation (1.11) becomes

$$\|\hat{u} - u^*\|_\infty \leq \frac{1}{1 - L\,|\Omega|} \mathcal{E}^{QUAD}(u^*) \tag{1.12}$$

Proof. Using algebraic manipulations we have the following inequalities:

$$\begin{aligned}
\|\tilde{u} - u^*\|_\infty &= \max_{t\in\Omega} \left| \cancel{f(t)} + \sum_{i=0}^{p} w_i \, K(t, x_i, u_i)) - \cancel{f(t)} - \int_\Omega K(t, x, u^*(x))\, dx \right| \\
&\qquad \textit{by adding and subtracting } \sum_{i=0}^{p} w_i \, K(t, x_i, u^*(x_i)) \\
&\leq \mathcal{E}^{\mathrm{QUAD}}(u^*) + \max_{t\in\Omega} \left| \sum_{i=0}^{p} w_i \Big(K(t, x_i, u_i) - K(t, x_i, u_i^*) \Big) \right| \\
&\leq \mathcal{E}^{\mathrm{QUAD}}(u^*) + \max_{t\in\Omega} \left\{ \sum_{i=0}^{p} w_i \Big| K(t, x_i, u_i) - K(t, x_i, u_i^*) \Big| \right\} \\
&\leq \mathcal{E}^{\mathrm{QUAD}}(u^*) + L \sum_{i=0}^{p} w_i \, |u_i - u_i^*| \\
&\leq \mathcal{E}^{\mathrm{QUAD}}(u^*) + L \underbrace{\left(\sum_{i=0}^{p} w_i \right)}_{|\Omega|} \|\mathbf{u} - \mathbf{u}^*\|_\infty
\end{aligned}$$

These inequalities prove the first error estimation (1.11). If we assume that $\mathbf{u} \equiv \mathbf{u}^*$ we can use the inequality (1.9), $\|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty \leq \frac{1}{1-L\,|\Omega|} \mathcal{E}^{\mathrm{QUAD}}(u^*)$ in order to obtain the second error estimation (1.12). □

The inequality (1.12) allows us to state that the error of approximation (1.6) is controlled by the error of quadrature rule and by the nearness of $L\,|\Omega|$ to 1. The inequality (1.11) is useful[5] when we obtain a numerical solution $\mathbf{u}$ of nonlinear system (1.7): if we are sure about the goodness of quadrature rule then we can estimate the error of approximation (1.10) by only comparing it to the value of $u^*(\cdot)$ on quadrature nodes and the vector $\mathbf{u}$.

Usually we don't know the solution u^*, therefore we need an estimation of the error to obtain a stopping criterion. We propose a posteriori estimation of the error by residual of a solution $\mathbf{u}$. In order to define the residual, we introduce the operator $\mathbf{T} : \mathbb{R}^{p+1} \to \mathbb{R}^{p+1}$ given by the relation (1.7),

$$(\mathbf{T}[\mathbf{u}])_k := f_k + \sum_{i=0}^{p} w_i\, \mathbf{K}_{k,i}(u_i) \qquad \text{for } k = 0, \ldots, p \tag{1.13}$$

It's clear that the residual of nonlinear system (1.7) relative to the solution $\mathbf{u}$ is the vector $\mathbf{u} - \mathbf{T}[\mathbf{u}]$. We also observe that if $K(\cdot,\cdot,\cdot)$ satisfies the condition $ii)$ of the Theorem 1.2 then we have[6] the same Lipschitz condition for the operator $\mathbf{T}$:

$$\|\mathbf{T}[\mathbf{u}] - \mathbf{T}[\mathbf{v}]\|_\infty \leq L\,|\Omega|\; \|\mathbf{u} - \mathbf{v}\|_\infty \qquad \forall \mathbf{u}, \mathbf{v} \in [-\zeta, \zeta]^{p+1} \subset \mathbb{R}^{p+1}$$

Thanks to this notation we can propose some bounds for the error based on the residual, $\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty$. In the following proposition we present it for the error of $\mathbf{u}$ with respect to the solution of nonlinear system (1.7).

Proposition 1.2. *We assume that the hypotheses $i) - iv)$ of the Theorem 1.2 hold. If $\mathbf{u} \in \widetilde{M}$ and $\bar{\mathbf{u}}$ is the solution of nonlinear system (1.7), we have the following bounds of the error between $\mathbf{u}$ and $\bar{\mathbf{u}}$ with respect to the $\|\cdot\|_\infty$ norm of the vector space $\mathbb{R}^{p+1}$:*

$$\frac{1}{1 + L\,|\Omega|}\, \|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty \leq \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty \leq \frac{1}{1 - L\,|\Omega|}\, \|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty \tag{1.14}$$

Proof. We show the inequalities separately. The first inequality is verified by the following algebraic steps:

$$\begin{aligned} \|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty &\leq \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty + \|\mathbf{T}[\bar{\mathbf{u}}] - \mathbf{T}[\mathbf{u}]\|_\infty \\ &\leq \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty + L\,|\Omega|\, \|\bar{\mathbf{u}} - \mathbf{u}\|_\infty \end{aligned}$$

In order to prove the second inequality, we show that

$$\begin{aligned} \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty &= \|\mathbf{u} - \mathbf{T}[\bar{\mathbf{u}}]\|_\infty \\ &\leq \|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty + \|\mathbf{T}[\mathbf{u}] - \mathbf{T}[\bar{\mathbf{u}}]\|_\infty \\ &\leq \|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty + L\,|\Omega|\, \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty \end{aligned}$$

[5] Thanks to this observation, when we want present the numerical result of our example, we will propose the behaviour of $\|\mathbf{u}^{(n)} - \mathbf{u}^*\|_\infty$ where $\{\mathbf{u}^{(n)}\}_{n\in\mathbb{N}}$ is the sequence given by our iterative method.

[6] For a proof of this property, see (1.17) in page 18.

and finally we carry the residual term to the left,

$$(1 - L\,|\Omega|)\ \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty \leq \|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty$$

since $L\,|\Omega| < 1$, we can divide for $(1 - L\,|\Omega|) > 0$ to obtain the thesis. □

Now, we want to prove a similar relation for the error with respect to the solution of Urysohn integral equation (1.1) which, as we have already seen in previous page, is important for our estimation (1.11).

Corollary 1.1. *We assume that the hypotheses $i) - iv)$ of the Theorem 1.2. If $\mathbf{u} \in \widetilde{M}$ and let $\mathbf{u}^* \in \widetilde{M}$ be the evalutations of the solution of Urysohn integral equation, u. We have the following estimation of the error between $\mathbf{u}$ and $\mathbf{u}^*$ with respect to the $\|\|_\infty$ norm of vector space $\mathbb{R}^{p+1}$:*

$$\frac{\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty}{1 + L\,|\Omega|} - \frac{\mathcal{E}^{QUAD}(u^*)}{1 - L\,|\Omega|} \leq \|\mathbf{u} - \mathbf{u}^*\|_\infty \leq \frac{\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty}{1 - L\,|\Omega|} + \frac{\mathcal{E}^{QUAD}(u^*)}{1 - L\,|\Omega|} \quad (1.15)$$

Proof. For the error estimations (1.9) and (1.14), we have:

$$\|\mathbf{u} - \mathbf{u}^*\|_\infty \leq \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty + \|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty \leq \frac{\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty}{1 - L\,|\Omega|} + \frac{\mathcal{E}^{\mathrm{QUAD}}(u^*)}{1 - L\,|\Omega|}$$

$$\|\mathbf{u} - \mathbf{u}^*\|_\infty \geq \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty - \|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty \geq \frac{\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty}{1 + L\,|\Omega|} - \frac{\mathcal{E}^{\mathrm{QUAD}}(u^*)}{1 - L\,|\Omega|}$$

□

As we have seen above, we want to consider the residual, $\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty$, instead of $\|\mathbf{u} - \mathbf{u}^*\|_\infty$ because we don't know the exact solution u^*. If we assume that $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ has the same order of magnitude as machine precision and the scalar $L\,|\Omega|$ is not too close to 1, then $\|\mathbf{u} - \mathbf{u}^*\|_\infty$ is comparable to $\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty$ according to the previous corollary. Otherwise if $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ is not negligible, we observe from the relation coming from (1.15)

$$\|\mathbf{u} - \mathbf{u}^*\|_\infty \geq \|\bar{\mathbf{u}} - \mathbf{u}^*\|_\infty - \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty \geq \frac{\mathcal{E}^{\mathrm{QUAD}}(u^*)}{1 + L\,|\Omega|} - \frac{\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty}{1 - L\,|\Omega|} \quad (1.16)$$

that $\|\mathbf{u} - \mathbf{u}^*\|_\infty$ has the same order of magnitude as $\mathcal{E}^{\mathrm{QUAD}}(u^*)$, even if the term $\|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty$ converges to machine precision.

1.3 Picard iteration to solve nonlinear systems

As made by Brezinski and Redivo-Zaglia in [10], we introduce Picard iteration to solve the non-linear system (1.7). In the current section we want to propose an analytic study where we deepen the existence and the uniqueness of the solution and the convergence behaviour.

Our presentation is based on the Banach Contraction Mapping Principle. We consider the Banach space $E = \mathbb{R}^{p+1}$ with the norm $\|\cdot\|_\infty$, the subset $\widetilde{M} = [-\zeta, \zeta]^{p+1} \subset E$, and the operator $\mathbf{T} : \mathbb{R}^{p+1} \to \mathbb{R}^{p+1}$ given by (1.7)

$$(\mathbf{T}[\mathbf{u}])_k = f_k + \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i), \quad k = 0, \ldots, p$$

If we assume the hypothesis *ii*) of the Theorem 1.2, the operator $\mathbf{T}[\cdot]$ is a contractor operator because for all $\mathbf{u} = (u_0, \ldots, u_p), \mathbf{v} = (v_0, \ldots, v_p) \in \widetilde{M}$ we have

$$\begin{aligned}
\|\mathbf{T}[\mathbf{u}] - \mathbf{T}[\mathbf{v}]\|_\infty &= \max_{k=0,\ldots,p} |(\mathbf{T}[\mathbf{u}])_k - |(\mathbf{T}[\mathbf{v}])_k| \\
&= \max_{k=0,\ldots,p} \left| \sum_{i=0}^{p} w_i \Big(K(x_k, x_i, u_i) - K(x_k, x_i, v_i) \Big) \right| \\
&\leq \max_{k=0,\ldots,p} \left\{ \sum_{i=0}^{p} w_i \left| K(x_k, x_i, u_i) - K(x_k, x_i, v_i) \right| \right\} \\
&\leq \max_{k=0,\ldots,p} \left\{ L \sum_{i=0}^{p} w_i \left| u_i - v_i \right| \right\} \\
&\leq \max_{k=0,\ldots,p} \left\{ L \sum_{i=0}^{p} w_i \right\} \|\mathbf{u} - \mathbf{v}\|_\infty \\
&= L|\Omega| \, \|\mathbf{u}_i - \mathbf{v}_i\|_\infty
\end{aligned} \tag{1.17}$$

In order to apply the BCM principle we have to be sure that the restriction of $\mathbf{T}[\cdot]$ to the subset M acts on M, $\mathbf{T}[M] \subset [M]$. In section §1.1 eqn. (1.5) we assume that

$$\max_{t \in \Omega, \, u \in C(\Omega; [-\zeta, \zeta])} \left| f(t) + \int_\Omega K(t, x, u(x)) \, dx \right| \leq \zeta$$

to prove $\mathfrak{T}[M] \subset M$ but in general cases this is not sufficient to state that $\mathbf{T}[\widetilde{M}] \subset \widetilde{M}$ because:

$$\begin{aligned}
\max_{k=0,\ldots,p, \mathbf{u} \in \widetilde{M}} |\mathbf{T}[\mathbf{u}]| &= \max_{k=0,\ldots,p, \mathbf{u} \in \widetilde{M}} \left| f_k + \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i) \right| \\
&\leq \max_{t \in \Omega, \, u \in C(\Omega; [-\zeta, \zeta])} \left| f(t) + \int_\Omega K(t, x, u(x)) \, dx \right| + \mathcal{E}^{\mathrm{QUAD}}(u)
\end{aligned}$$

For example if the maximizer function u in (1.5) is a constant function (like Example 1.1) then the term $\mathcal{E}^{\mathrm{QUAD}}(u)$ is equal to[7] 0.

Observation 1.4. *Since the condition v) of the Theorem 1.3 (we will read it in this page) is difficult to study from a practical point of view, we will introduce a check at each iteration n for checking if* $\mathbf{u}^{(n)} \in \widetilde{M}$. *Hence this check will be the discriminatory factor: if exists an n such that* $\mathbf{u}^{(n)} \notin \widetilde{M}$, *then we deduce that the fifth condition of the Theorem isn't verified.*

In the following, we recall the previous Theorem 1.2 and we define the Picard iteration for the non-linear system (1.7).

Theorem 1.3. *We assume that the conditions i) − iv) of the Theorem 1.2 hold. By the Nyström Method we approximate the Urysohn integral equation to a quadrature formula* $\{(x_k, w_k)\}_{k=0,\dots,p}$. *We need to solve the nonlinear system* (1.7)

$$u_k = f_k + \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i) \quad \textit{for } k = 0, \dots, p$$

i.e. $\mathbf{u} = \mathbf{T}[\mathbf{u}]$. *If we assume that* $\mathbf{T}[\widetilde{M}] \subset \widetilde{M}$, *i.e.*

v) $\max_{k=0,\dots,p,\|\mathbf{u}\|_\infty \leq \zeta} |f_k + \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i)| < \zeta$

then there is a unique solution $\bar{\mathbf{u}}$ *for the nonlinear system* (1.7).
If $\mathbf{u}^{(0)}$ *is an arbitrary element of* $\mathbb{R}^{p+1}$ *that satisfies the inequality* $\left\|\mathbf{u}^{(0)}\right\|_\infty \leq \zeta$ *then the sequence* $\mathbf{u}^{(n)}$ *given by*

$$\mathbf{u}^{(n+1)} := \mathbf{T}[\mathbf{u}] \quad \textit{i.e.} \quad u_k^{(n+1)} = f_k + \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i^{(n)}) \ \textit{ for } K = 0, \dots, p \tag{1.18}$$

converge to $\bar{\mathbf{u}}$, *and the sequence identified by* (1.18) *is known as* ***Picard iteration***.

Moreover we can observe the following corollary that shows the decrease of the residual and of the error of every Picard iteration.

Corollary 1.2. *We assume that the hypotheses of the Theorem 1.3 hold. If we consider the Picard iteration,* $\mathbf{u}^{(n)} = \mathbf{T}[\mathbf{u}]$, *we have the following inequality for the residual*

$$\left\|\mathbf{u}^{(n+1)} - \mathbf{T}[\mathbf{u}^{(n+1)}]\right\|_\infty \leq L|\Omega| \, \left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty \tag{1.19}$$

and the following one for the error

$$\left\|\mathbf{u}^{(n+1)} - \mathbf{u}^*\right\|_\infty \leq L|\Omega| \, \left\|\mathbf{u}^{(n)} - \mathbf{u}^*\right\|_\infty \tag{1.20}$$

[7] It's true if quadrature formula has at least order 1.

Proof. By the last hypothesis of the Theorem 1.3 we know that $\mathbf{T}[\widetilde{M}] \subset \widetilde{M}$ therefore the Lipschitz condition is always verified through the operator images. Then we have:

$$\left\|\mathbf{u}^{(n+1)} - \mathbf{T}[\mathbf{u}^{(n+1)}]\right\|_\infty = \left\|\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{T}\left[\mathbf{T}[\mathbf{u}^{(n)}]\right]\right\|_\infty \leq L|\Omega| \left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty$$
$$\left\|\mathbf{u}^{(n+1)} - \mathbf{u}^*\right\|_\infty = \left\|\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{T}[\mathbf{u}^*]\right\|_\infty \leq L|\Omega| \left\|\mathbf{u}^{(n)} - \mathbf{u}^*\right\|_\infty$$

□

The Picard iteration represents a common numerical method to find a fixed point of a contraction operator. But in the literature [5] are presented different relaxations that allow to determine the fixed points also for operators that are not contractive operators. They are based on a better asymptotic behaviour of a new operator $\widetilde{\mathbf{T}}$ than the original mapping $\mathbf{T}$. An example of this relaxation is

$$\widetilde{\mathbf{T}}_\alpha[\mathbf{u}] := \mathbf{u} + \alpha\left(\mathbf{T}[\mathbf{u}] - \mathbf{u}\right)$$

If we iterate the relaxation for a fixed α, we obtain the so called *Krasnoselskij Iteration* (see [5, §3]). Otherwise if we choose the $\alpha^{(n)}$ parameter at each n-th iteration we obtain a more versatile relaxation called *Mann Iteration* (see [5, §4]) or **Relaxed Picard Iteration**:

$$\mathbf{u}^{(n+1)} := \mathbf{u}^{(n)} + \alpha^{(n)}\left(\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{u}^{(n)}\right) \tag{1.21}$$

In most of the numerical examples that we will show in this work we assume that contraction property is strictly satisfied. However we will also present some numerically interesting examples that do not satisfy such property (we will meet one of these examples at the end of this chapter). It is not the focus of this thesis to generalize the assumptions presented in [10], but to improve the performance of extrapolation by deepening the analytical aspects and experimenting it on different cases.

The Relaxed Picard Iteration has inspired us to face this problem from a different perspective. We can interpret the nonlinear system(1.7) as a bound constrained optimization problem where objective function is nonlinear:

$$\begin{aligned} &\min_{\mathbf{u}\in\mathbb{R}^{p+1}} \|\mathbf{u} - \mathbf{T}[\mathbf{u}]\|_\infty \\ &\quad s.t. -\zeta \leq \mathbf{u}_k \leq \zeta \quad k = 0, \ldots, p \end{aligned} \tag{1.22}$$

The inequalities (1.14) exclude the possibility to find an element with a low residual and a high error. Therefore we can formulate a global optimal search algorithm that in every iteration builds a vector $\mathbf{u}^{(n+1)}$ that has a lower residual than the previous one. Most of the optimization algorithms use the derivative

of the objective function $\mathbf{F}[\cdot]$ in order to determinate a *descent direction*, i.e. a vector $\mathbf{d}^{(n+1)}$ such that

$$\mathbf{F}[\mathbf{u}^{(n)} + \alpha^{(n+1)}\mathbf{d}^{(n+1)}] < \mathbf{F}[\mathbf{u}^{(n)}]$$

for some positive scalar $\alpha^{(n+1)} > 0$ called *step size*.
Unfortunately the objective function of (1.22) is not regular hence we have to determinate a descent direction through another criterion. The inequality (1.19) defined in Corollary 1.2 suggests that we can consider the vector $\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{u}^{(n)}$ as descent direction and we obtain an iteration algorithm given by

$$\mathbf{u}^{(n+1)} := \mathbf{u}^{(n)} + \alpha^{(n+1)} \left(\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{u}^{(n)}\right)$$

which coincides with Relaxed Picard Iteration (1.21). We have to prove that the vector $\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{u}^{(n)}$ is a descent direction.

Proposition 1.3. *We assume that the hypotheses of the Theorem 1.3 hold. We consider the Relaxed Picard iteration,* $\mathbf{u}^{(n+1)} := \mathbf{u}^{(n)} + \alpha^{(n+1)} \left(\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{u}^{(n)}\right)$. *If the step size* $\alpha^{(n)} > 0 \, \forall n$, *then we have*

$$\left\|\mathbf{u}^{(n+1)} - \mathbf{T}[\mathbf{u}^{(n+1)}]\right\|_\infty \leq \left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty$$

Proof. Adding end subtracting the term $\mathbf{T}[\mathbf{u}^{(n)}]$ we have:

$$\left\|\mathbf{u}^{(n+1)} - \mathbf{T}[\mathbf{u}^{(n+1)}]\right\|_\infty \leq \left\|\mathbf{u}^{(n+1)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty + \left\|\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{T}[\mathbf{u}^{(n+1)}]\right\|_\infty$$

We study the two terms separately

$$\begin{aligned}
\left\|\mathbf{u}^{(n+1)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty &= \left\|\mathbf{u}^{(n)} + \alpha^{(n+1)} \left(\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{u}^{(n)}\right) - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty \\
&= (1 - \alpha^{(n+1)}) \left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty \\
\left\|\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{T}[\mathbf{u}^{(n+1)}]\right\|_\infty &\leq L\,|\Omega| \left\|\mathbf{u}^{(n)} - \mathbf{u}^{(n+1)}\right\|_\infty \\
&= L\,|\Omega| \left\|\cancel{\mathbf{u}^{(n)}} - \cancel{\mathbf{u}^{(n)}} + \alpha^{(n+1)} \left(\mathbf{T}[\mathbf{u}^{(n)}] - \mathbf{u}^{(n)}\right)\right\|_\infty \\
&= L\,|\Omega|\,\alpha^{(n+1)} \left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}\right\|_\infty
\end{aligned}$$

Therefore we have

$$\left\|\mathbf{u}^{(n+1)} - \mathbf{T}[\mathbf{u}^{(n+1)}\right\|_\infty \leq \left(1 - \alpha^{(n+1)} + L\,|\Omega|\,\alpha^{(n+1)}\right) \left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}\right\|_\infty$$

and the factor $1 - \alpha^{(n+1)} + L\,|\Omega|\,\alpha^{(n+1)}$ is less than 1 if, and only if,

$$\alpha^{(n+1)}(1 - L\,|\Omega|) > 0$$

Since $L\,|\Omega| < 1$ by hypothesis, we demonstrate the thesis. □

1.4 Numerical results

We would like to close this chapter presenting some numerical examples of the Relaxed Picard iteration method (RPIM). In particular we focus on RPIM with fixed step size[8] $\alpha^{(n)} = \alpha > 0 \, \forall n$ from which we want to draw some empirical considerations useful for what we will show in next chapters. We can synthesize this algorithm with the pseudo-code scheme below where in input there are the starting vector $\mathbf{u}^{(0)}$, the vector operator $\mathbf{T}$ given by (1.13), the step size α, the end ζ[9] of the interval $\widetilde{M} = [-\zeta, \zeta]$, the tolerance Tol, and the maximum number of iterations Max_Iter for the stopping criterion[10].

The conditions of the Theorem 1.3 guarantee that RPIM converges to the unique solution identified by the theorem as we have seen before. But the condition v), which is dependent on the quadrature formula, is difficult to study from a theoretical point of view and hence in 9-th row of the Algorithm 1 we implement the control of which we have talked about in the Observation 1.4. In this way, if the user provides $\mathbf{u}^{(0)}$ belonging to the confidential region $\widetilde{M}$, the Algorithm gives a warning if the current iteration doesn't belong to this region.

Algorithm 1 Relaxed Picard Iteration Method (RPIM)

1: **procedure** RPIM($\mathbf{u}^{(0)}$, $\mathbf{T}$, α, ζ, Tol, Max_Iter)
2: **res** $\leftarrow$ zeros-array of Max_Iter dimension
3: **im** $\leftarrow \mathbf{T}[\mathbf{u}^{(0)}]$
4: res[0] $\leftarrow \left\| \mathbf{u}^{(0)} - \mathbf{im} \right\|_\infty$
5: $n \leftarrow 0$
6: **while** res[n] $>$ Tol **and** $n \leq$ Max_Iter **do**
7: $n \leftarrow n + 1$
8: $\mathbf{u}^{(n)} \leftarrow (1 - \alpha)\, \mathbf{u}^{(n-1)} + \alpha\, \mathbf{im}$
9: **if** $\left\| \mathbf{u}^{(n)} \right\|_\infty > \zeta$ **then**
10: **Warning**: "the n-th solution is outside of the conditions"
11: **end if**
12: **im** $\leftarrow \mathbf{T}[\mathbf{u}^{(n)}]$
13: res[n] $\leftarrow \left\| \mathbf{u}^{(n)} - \mathbf{im} \right\|_\infty$
14: **end while**
15: **end procedure**

This algorithm is implemented by us in `RPIM.m` MATLAB's file and we use it in order to determine the numerical considerations that follow.

[8] As we have already seen, this method is called *Krasnoselskij Iteration Method*, see [5, §3].

[9] In the examples where the condition of Theorem 1.3 isn't satisfied, we set $\zeta = +\infty$.

[10] In the experiments of the next pages we will set tolerance to $5 \cdot 10^{-15}$, and the maximum number of iterations to 50.

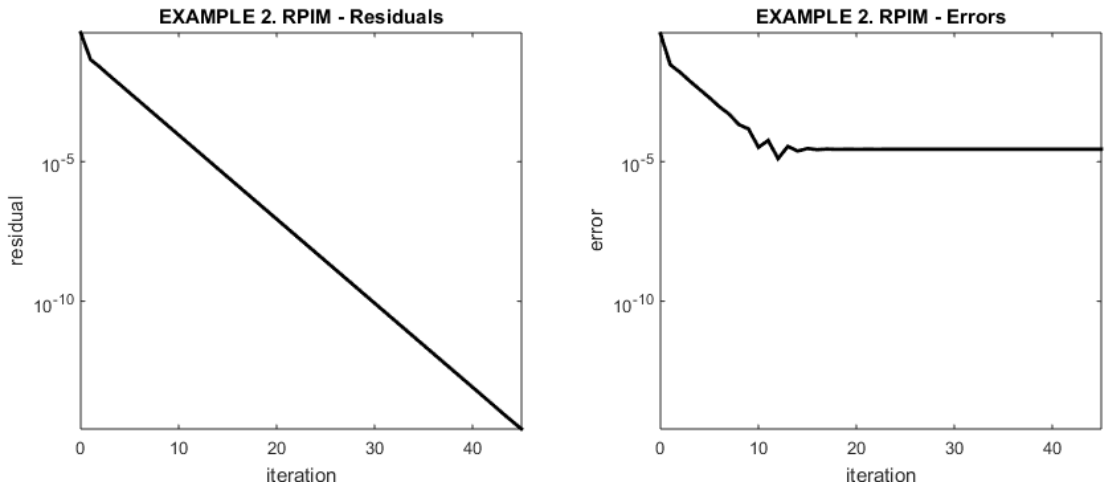

Figure 1.1: Example 2 - Residuals and errors behaviours of RPIM with $\alpha = 1$ using trapezoidal rule with $p = 21$.

Example 2

Now we consider the Example 2 of [10] (for this reason we prefer to keep the same number). We want to deepen from the numerical point of view the integral equation studied in Observation 1.1,

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

We proved that the conditions of Theorem 1.3 are satisfied if $\widetilde{M} = [-\zeta, \zeta]$ with $\zeta \in \left(\frac{1}{4}, \frac{4}{3\sqrt{2\pi}}\right)$. Therefore if we start with a vector $\mathbf{u}^{(0)}$ that satisfies the inequality $\left\|\mathbf{u}^{(0)}\right\| \leq \frac{4}{3\sqrt{2\pi}}$ we are sure that Picard iteration converge to the solution of the non-linear system and we start from $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$. With notation $\mathbf{v} \equiv c$ we indicate the vector $\mathbf{v}$ having all component equal to the scalar c.
If we want that the approximation (1.10)

$$\tilde{u}^{(n)}(t) := f(t) + \sum_{i=0}^{p} w_i\, K(t, x_i, u_i^{(n)}) \quad \text{for } t \in \Omega$$

converges also to the solution $u_2^*(t) = -\frac{1}{5}\cos\left(\frac{\pi t}{4}\right)$, it is necessary that in the quadrature rule the term $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ is negligible. In fact in (1.16) we proved that the error of $\mathbf{u}^{(n)}$ with respect to $\mathbf{u}^*$ satisfies the inequality:

$$\left\|\mathbf{u}^{(n)} - \mathbf{u}^*\right\|_\infty \geq \frac{\mathcal{E}^{\mathrm{QUAD}}(u^*)}{1 + L\,|\Omega|} - \frac{\left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty}{1 - L\,|\Omega|}$$

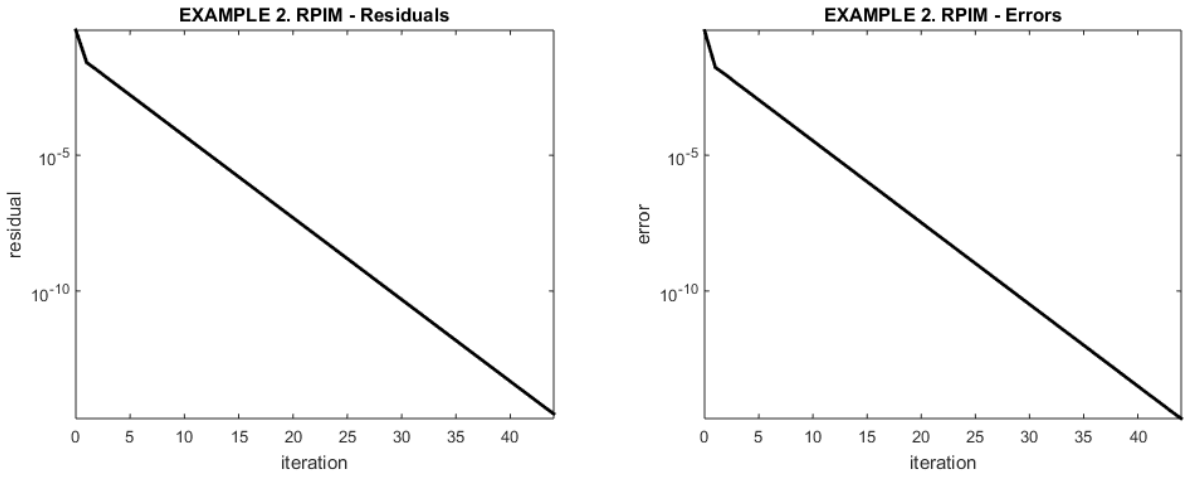

Figure 1.2: Example 2 - Residuals and errors behaviours of RPIM with $\alpha = 1$ using the Gaussian formula with $p = 8$.

Hence we expect that the error follows the behaviour of the residuals as long as its magnitude is greater than the term $\mathcal{E}^{\mathrm{QUAD}}(u^*)$. We investigate this argumentation from a practical point of view comparing two different quadrature formulas.

In the first case we use a trapezoidal rule with $p = 21$ i.e. 22 nodes: this quadrature formula is the same used in Brezinski and Redivo-Zaglia [10]. In Figure 1.1 we show the residuals $\left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty$ and the errors $\left\|\mathbf{u}^{(n)} - \mathbf{u}^*\right\|_\infty$ where $\mathbf{u}^*$ rappresents the evalutations of u_2^* on the quadrature nodes. We can observe that the behaviour of residuals and errors is the same until 7-th iteration, after that we have a steady-state error of magnitude order 10^{-5}. The same phenomenon is showed in Figure 3 of [10], where the error is in a steady-state of magnitude order 10^{-3}. The difference of magnitude order is justified by the fact that Brezinski and Redivo-Zaglia start from a vector outside of the region $\widetilde{M}$ and compare the error with $u_1^*(t) = \cos\left(\frac{\pi t}{4}\right)$ instead the solution $u_2^*(t) = \frac{1}{5}\cos\left(\frac{\pi t}{4}\right)$ which image belonging to the region $\widetilde{M}$.

In the second case we consider the set of Legendre-Gauss quadrature formulas that exhibits a better asymptotic behaviour. In our notation the parameter p indicates the unique Legendre-Gauss quadrature formula which has $p + 1$ nodes. In Figure 1.2 we show the error and residual as before using the Legendre-Gauss quadrature formula with $p = 8$. In this case we don't observe the same phenomenon emerged in the previous setting. Therefore we expect that the first quadrature formula has the term $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ of magnitude 10^{-5} while the second quadrature has it negligible. In Figure 1.3 we show the term $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ of both quadrature formulas for every parameter p (the number of intervals in which we divide the domain $[a, b]$) varying on natural number from 1 to 50. It is determined

by the definition (1.8) using the exact solution u_2^*,

$$\mathcal{E}^{\mathrm{QUAD}}(u^*) = \max_{t\in\Omega} \left| u_2^*(t) - f(t) - \sum_{i=0}^{p} w_p \, K(t, x_i, u_2^*(x_i)) \right|$$

In this way for a set of quadrature formulas we can determine the parameter p necessary to ensure that the order of magnitude of $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ is close to the machine magnitude order for each example. As we can see in the figure in the following page, in the Example 2 we chose the Legendre-Gauss quadrature formula with $p = 8$ for the reason we have just exposed. Unless we want to compare the results of other works, we will always prefer to use the family of Legendre-Gauss quadrature formulas in order to avoid the previous phenomenon. For easy of reference we indicate the Legendre-Gauss by "the" Gaussian quadrature formula as often happens in literature.

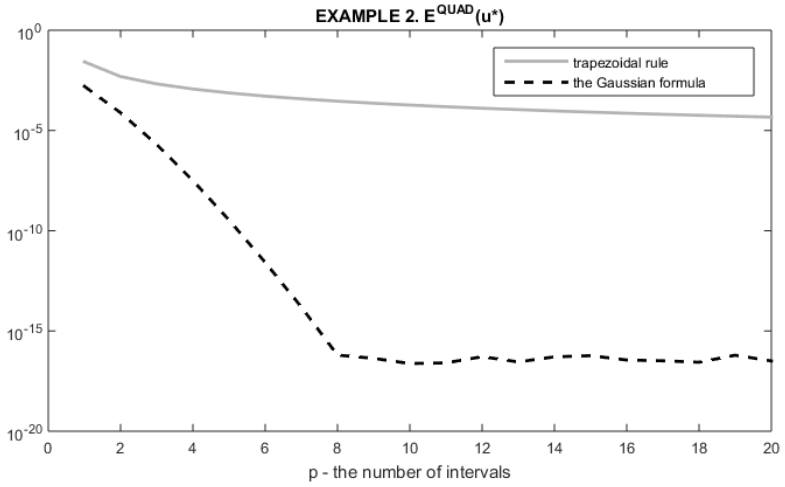

Figure 1.3: Behaviour of $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ by using the Gaussian formula and trapezoidal rule varying on parameter p.

Example 3

Finally we want to propose a case in which relaxation is necessary in order to achieve convergence. In this regard, we have to study an Urysohn integral equation where the $\mathbf{T}$ is not a contraction operator as we discussed before.

We consider the Example 5.3 of Borzabadi, Fard's article [6, p. 453], corrected in [10, p. 19] by Brezinski, Redivo-Zaglia, where we have the integral equation

$$u(t) = -\int_0^1 (x+t)e^{u(x)}\,dx + e\,t + 1 \tag{1.23}$$

whose solution is $u^*(t) = t$.

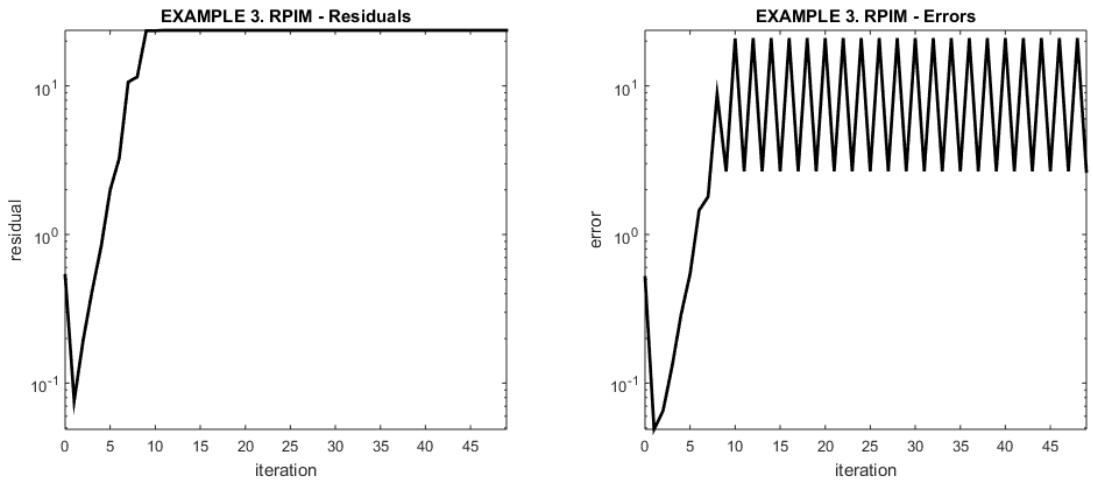

Figure 1.4: Example 3 - Residuals and errors behaviours of RPIM with $\alpha = 1$.

In the numerical example we consider the Gaussian quadrature formula with $p = 5$ and we start from $\mathbf{u}^{(0)} \equiv 1$. The authors omitted the fact that the operator is expansive because it's exponential. Applying the simple Picard Iteration Method we can see what happens in Figure 1.4 where we show the residuals $\left\|\mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}]\right\|_\infty$ and the errors $\left\|\mathbf{u}^{(n)} - \mathbf{u}^*\right\|_\infty$. We note that the method diverges as we expected.

On the contrary, in Figure 1.5 we note that it converges by using a PCIM with $\alpha = 0.55$ and starting from the same solution $\mathbf{u}^* \equiv \frac{2}{3}$. This shows how the choice of a lower α value for RPIM is decisive for convergence in cases where conditions of Theorem 1.3 are not satisfied.

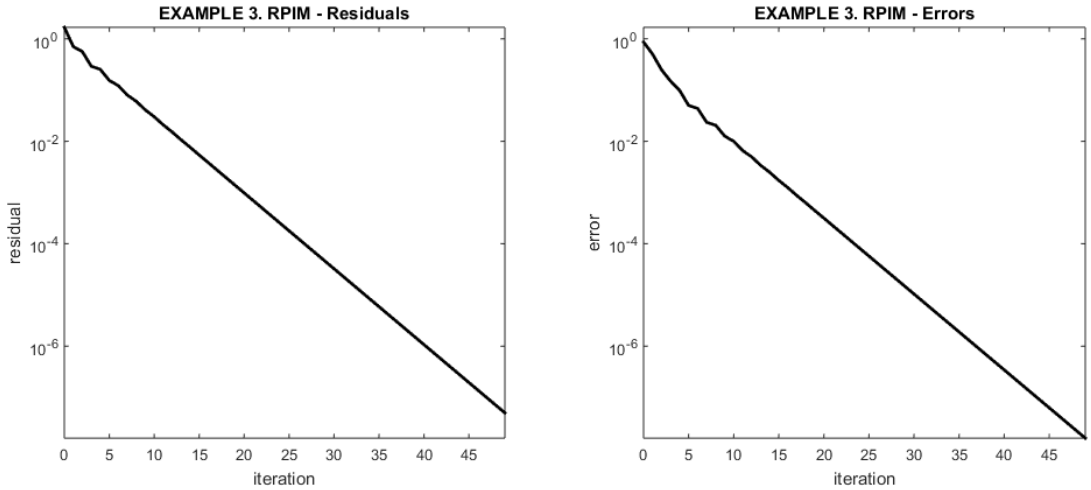

Figure 1.5: Example 3 - Residuals and errors behaviours of RPIM with $\alpha = 0.55$.

Chapter 2

Extrapolation Methods for integral equations

As we have seen in the previous chapter, if we assume some hypotheses[1] the Urysohn integral equations

$$u(t) = f(t) + \int_{\Omega} K(t, x, u(x))\, dx, \quad \text{for } t \in \Omega$$

have a unique continuous solution, u^*, which satisfies the inequality $\|u^*\|_\infty \leq \zeta$ where ζ is a parameter appearing in the hypotheses. Moreover when $\{(x_i, w_i)\}_{i=0,\dots,p}$ are the nodes with the associated positive weights of an appropriate[2] quadrature formula, we have a sequence of numerical approximations given by $\tilde{u}^{(n)}(t) = f(t) + \sum_{i=0}^{p} w_i\, K(t, x_i, u_i^{(n)})$ where $\mathbf{u}^{(n)} = (u_0^{(n)}, \dots, \mathbf{u}_p^{(n)})$ is obtained by the Relaxed Picard iteration method

$$u_k^{(n+1)} = u_k^{(n)} + \alpha^{(n+1)} \left(f_k + \sum_{i=0}^{p} w_i\, \mathbf{K}_{k,i}(u_i^{(n)}) - u_k^{(n)} \right) \quad k = 0, \dots, p \tag{2.1}$$

for a starting element $\mathbf{u}^{(0)}$ which satisfies the inequality $\left\|\mathbf{u}^{(0)}\right\|_\infty \leq \zeta$ and an arbitrary step size $\alpha^{(n+1)}$ that determines a residual reduction.
However in the numerical results proposed in §1.4 we saw that the convergence of this numerical method sometimes is slow. The goal of this chapter is to combine the RPIM with an acceleration method based on extrapolation method. In the Brezinski, Redivo-Zaglia's article [10] this idea has reached good results but they are flawed by the use of an inefficient quadrature formula like trapezoidal rule[3]. Furthermore, in order to improve the performance of extrapolation method we investigate an adaptive strategy for the choice of $\alpha^{(n)}$ instead of keeping it fixed as

[1] See the Theorem 1.3.

[2] See the Corollary 1.2 and the Observation 1.3

[3] See the page 23 of this thesis where we studied the comparison of the effect in the use of the rule of trapezoidal with respect to the Gaussian formula for a basic Picard Iteration Method.

in the article [10]. We start recalling some definition and properties of the Shanks' transformation, one of the most known one used for accelerating a sequence of scalars, vectors, matrices, and tensors.

2.1 Shanks' transformation for scalar sequence

When a scalar sequence $(S_n)_{n\in\mathbb{N}} \subset \mathbb{R}$ is slowly converging to S, we can transform it into a new sequence which converges faster to the same limit. We say that a sequence $(T_n)_{n\in\mathbb{N}}$ accelerates the convergence of (S_n) if the sequence (T_n) converges to the same limit S of (S_n) and

$$\lim_{n\to+\infty} \frac{T_n - S}{S_n - S} = 0.$$

But Delahaye and Germain-Bonne [15] that there isn't any transformation able to accelerate all the possible converging sequences. We infer that every transformation is only able to accelerate the convergence of certain classes of sequence. Therefore when we consider an extrapolation method we have to check, if possible, that the sequence belongs to a class of that can be accelerated.

Clarified these fundamental elements, we present the **Aitken's Δ^2 process**, one of the most well-known sequence transformation (see [1]) given by the following expression:

$$T_n = S_n - \frac{(S_{n+1} - S_n)^2}{S_{n+2} - 2S_{n+1} + S_n} \tag{2.2}$$

According to the discussion above, we have to determinate the class of sequences to which this transformation converges to the same limit faster than the original sequence.

Proposition 2.1. *The Aitken's Δ^2 process accelerate the convergence of all the sequences for which $\exists\lambda \in (-1, 1)$ such that*

$$\lim_{n\to+\infty} \frac{S_{n+1} - S}{S_n - S} = \lambda \tag{2.3}$$

Proof. We drew inspiration from the proof in [26, p. 2].
For simplicity, we introduce the sequence $(\delta_n)_{n\in\mathbb{N}}$ given by

$$\delta_n := \frac{S_{n+1} - S}{S_n - S} - \lambda$$

For hypothesis (2.3) we know that $\lim_{n\to+\infty} \delta_n = 0$. We have also the following relation:

$$\begin{aligned} S_{n+1} &= S + \big(S_n - S\big)\big(\lambda + \delta_n\big) \\ S_{n+2} &= S + \big(S_{n+1} - S\big)\big(\lambda + \delta_{n+1}\big) = S + (S_n - S)(\lambda + \delta_n)(\lambda + \delta_{n+1}) \end{aligned}$$

Hence we have:

$$\begin{aligned} S_{n+1} - S_n &= -\big(S_n - S\big) + \big(S_n - S\big)\big(\lambda + \delta_n\big) = \big(S_n - S\big)\big(\lambda + \delta_b - 1\big) \\ S_{n+2} - S_{n+1} &= \big(S_{n+1} - S\big)\big(\lambda + \delta_{n+1} - 1\big) = \big(S_n - S\big)\big(\lambda + \delta_n\big)\big(\lambda + \delta_{n+1} - 1\big) \end{aligned} \tag{2.4}$$

Using the relations (2.4), we have:

$$\begin{aligned} \frac{T_n - S}{S_n - S} &= \frac{S_n - \frac{(S_{n+1}-S_n)^2}{S_{n+2}-2S_{n+1}+S_n} - S}{S_n - S} = 1 - \frac{\left(S_{n+1} - S_n\right)^2}{\left(S_{n+2} - 2S_{n+1} + S_n\right)\left(S_n - S\right)} \\ &= 1 - \frac{\left(S - S_n\right)^2\left(\lambda + \delta_n - 1\right)^2}{\left[\left(\lambda + \delta_n\right)\left(\lambda + \delta_{n+1} - 1\right) - \left(\lambda + \delta_n - 1\right)\right]\left(S_n - S\right)^2} \\ &= 1 - \frac{\left(\lambda + \delta_n - 1\right)^2}{\left(\lambda + \delta_n\right)\left(\lambda + \delta_{n+1}\right) - 2\left(\lambda + \delta_n\right) + 1} \end{aligned}$$

Taking the limit as $n \to +\infty$, we have

$$\begin{aligned} \lim_{n\to+\infty} \frac{T_n - S}{S_n - S} &= \lim_{n\to+\infty} 1 - \frac{\left(\lambda + \delta_n - 1\right)^2}{\left(\lambda + \delta_n\right)\left(\lambda + \delta_{n+1}\right) - 2\left(\lambda + \delta_n\right) + 1} \\ &= 1 - \frac{\left(\lambda - 1\right)^2}{\lambda^2 - 2\lambda + 1} \end{aligned}$$

If $\lambda \neq 1$ then the transformed sequence converges faster than S_n.

Moreover if $(S_n)_{n\in\mathbb{N}}$ converges to S then we must to have $|\lambda| \leq 1$.
In fact, if $|\lambda| > 1$, for the definition of limit $\lim_{n\to+\infty} \frac{|S_{n+1}-S|}{|S_n-S|}$ we have for all $\delta \in (1, |\lambda|)$ that there is $\bar{n}$ such that

$$|S_n - S| \geq \delta |S_{n-1} - S| \quad \forall n \geq \bar{n}$$

Hence, we have:

$$|S_n - S| \geq \delta^{n-\bar{n}} |S_{\bar{n}} - S|$$

from which we infer that $\lim_{n\to+\infty} |S_n - S| \geq +\infty$. □

An important definition in these methods is the **Kernel of a transformation**: $\mathcal{K}_T$ is the set of all the sequences $(S_n)_{n\in\mathbb{N}}$ which are transformed by the transformation T into a constant sequence, i.e. $\exists N, S$[4] such that $T_n = S$, $\forall n \geq N$ for some N.

Observation 2.1. *If the original sequence belongs to the kernel of the transformation then $T_n = S$ $\forall n$ by construction. Although it has not been proved yet, numerical experiments have always confirmed that the "closer" a sequence is to the kernel, the faster the transformed sequence will converge.*

[4]Usually S is the limit of the sequence but this is not always the case. For example if we consider the Aitken's Δ^2 process with $|\lambda| > 1$ the sequence doesn't converge and S is called its anti-limit.

The *explicit form* of the kernel of Δ^2 process corresponds to the sequences having the form

$$S_n = S + a\,\lambda^n, \quad n = 0, 1, \ldots$$

where a and λ are scalar such that $a \neq 0$ and $\lambda \notin \{0, 1\}$.
This expression gives the explicit form of the sequences belonging to the kernel. However for this process the kernel can be also given in an *implicit form* that is a relation which holds among consecutive terms of the sequence. In the case of Aitken's process, we have the following implicit form:

$$S_{n+1} - S = \lambda\,(S_n - S), \quad n = 0, 1, \ldots$$

In general, for a transformation T, the implicit form of the kernel consists in a relation among consecutive terms of the sequence, involving the unknown parameters $a_1, \ldots, a_p$ and S which must be satisfied $\forall n$, if and only if (S_n) belongs to the kernel $\mathcal{K}_T$ of the transformation T. Thus can rewrite the kernel of the Δ^2 process using this form:

$$a_0\,(S_n - S) + a_1\,(S_{n+1} - S) = 0, \quad n = 0, 1, \ldots \tag{2.5}$$

which $a_0\,a_1 \neq 0$ and $a_0 + a_1 \neq 0$.

The aim of this section is to present a more transformation generalizing the Aitken's Δ^2 process, called **Shanks' transformation** [27]. The kernel, given by (2.5), suggested to define a transformation whose kernel consists of sequences satisfying the homogeneous linear difference equation of order k:

$$a_0\,(S_n - S) + a_1\,(S_{n+1} - S) + \ldots + a_k\,(S_{n+k} - S) = 0, \quad n = 0, 1, \ldots \tag{2.6}$$

where the coefficients a_i are arbitrary constants independent of n such that $a_0\,a_k \neq 0$ and $a_0 + \cdots + a_k \neq 0$, and where the unknown S is the limit of (S_n) if it converges. Assuming that $a_0 + \cdots + a_k = 1$ we can directly compute S from (2.6):

$$S = a_0\,S_n + \ldots + a_k\,S_{n+k}, \quad n = 0, 1, \ldots \tag{2.7}$$

Let (S_n) be sequence belonging to the kernel, we can eliminate the dependence from the term S taking (2.7) for n and $n + 1$, subtracting the two relations and writing:

$$a_0\,\Delta S_n + \ldots + a_k\,\Delta S_{n+k} = 0 \quad n = 0, 1, \ldots \tag{2.8}$$

where we denote $\Delta S_i := S_{i+1} - S_i$, for all i.

In order to determinate the transformation related to (2.6), we associate it with a sequence (S_n) which doesn't belong to the kernel, the linear transformation given by the right part of (2.7). The unknown coefficients can be computed solving the system obtained writing (2.8) for $n, \ldots, n + k - 1$ together with the

condition $a_0 + \ldots + a_k = 1$. This linear system has $k+1$ equation and $k+1$ variables therefore it individuates almost one solution. This system can be solved also when the sequence (S_n) doesn't belong to the kernel and (2.7) define the so called Shanks' transformation where the coefficients and the linear combination will now depend on n and k, and for indicating this dependence the coefficients are now denoted $a_i^{(n,k)}$ for $i = 0, \ldots, k$.

In summary, the Shanks' transformation converts the (S_n) into the sequence

$$e_k(S_n) = a_0^{(n,k)} S_n + \ldots + a_k^{(n,k)} S_{n+k}, \quad k, n = 0, 1, \ldots \tag{2.9}$$

where the $a_i^{(n,k)}$ are solution of the linear system:

$$\begin{cases} a_0^{(n,k)} + \ldots + a_k^{(n,k)} = 1 \\ a_0^{(n,k)} \Delta S_n + \ldots + a_k^{(n,k)} \Delta S_{n+k} = 0 \\ \vdots \\ a_0^{(n,k)} \Delta S_{n+k-1} + \ldots + a_k^{(n,k)} \Delta S_{n+2k-1} = 0 \end{cases} \tag{2.10}$$

It is a generalization of the Aitken's Δ^2 process which corresponds to the case $k = 1$. The parameter k is related to the number of iterations used by the extrapolation method. It's intuitive to think that a larger k provides a better performance as long as the previous information is consistent. But as we will see in the next section these results have an instability due to cancellation errors. We will propose an implementation of a particular rule in order to reduce this phenomenon.

2.2 ε-algorithm and Cordellier's particular rule

The system (2.10) gives us the possibility to compute the coefficient of (2.9). Nevertheless we can compute the transformed sequence $(e_k(S_n))$ without establishing the coefficients as the following ratio of determinants by Cramers' rule:

$$e_k(S_n) = \frac{\begin{vmatrix} S_n & \ldots & S_{n+k} \\ \Delta S_n & \ldots & \Delta S_{n+k} \\ \vdots & & \vdots \\ \Delta S_{n+k-1} & \ldots & \Delta S_{n+2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \ldots & 1 \\ \Delta S_n & \ldots & \Delta S_{n+k} \\ \vdots & & \vdots \\ \Delta S_{n+k-1} & \ldots & \Delta S_{n+2k-1} \end{vmatrix}} \tag{2.11}$$

The computation of $e_k(S_n)$ given by (2.11) needs to evaluate two determinants dimension $k+1$, this means that we need $2(k+1)(k+1)!$ multiplications for each

$\varepsilon_{-1}^{(0)}=0$				
	$\varepsilon_0^{(0)}=S_0$			
$\varepsilon_{-1}^{(1)}=0$		$\varepsilon_1^{(0)}$		
	$\varepsilon_0^{(1)}=S_1$		$\varepsilon_2^{(0)}$	
$\varepsilon_{-1}^{(2)}=0$		$\varepsilon_1^{(1)}$		$\varepsilon_3^{(0)}$
	$\varepsilon_0^{(2)}=S_2$		$\varepsilon_2^{(1)}$	
$\varepsilon_{-1}^{(3)}=0$		$\varepsilon_1^{(2)}$		$\varepsilon_3^{(1)}$
	$\varepsilon_0^{(3)}=S_3$		$\varepsilon_2^{(2)}$	
$\varepsilon_{-1}^{(4)}=0$		$\varepsilon_1^{(3)}$		
	$\varepsilon_0^{(4)}=S_4$			
$\varepsilon_{-1}^{(5)}=0$				

Table 2.1: The ε-*array* data structure.

iterations and in Numerical Analysis it is unfeasible. But since our determinants have some special structures, it's possible to obtain some rules to compute recursively $e_k(S_n)$. One of the most important scalar extrapolation algorithm is the **ε-algorithm** proposed by Wynn [29]. It computes the quantities $\varepsilon_k^{(n)}$ given by

$$\begin{cases} \varepsilon_{-1}^{(n)}=0, \quad \varepsilon_0^{(n)}=S_n, & n=0,1,\ldots \\ \varepsilon_{k+1}^{(n)}=\varepsilon_{k-1}^{(n+1)}+\dfrac{1}{\varepsilon_k^{(n+1)}-\varepsilon_k^{(n)}}, & k,n=0,1,\ldots \end{cases} \tag{2.12}$$

The numbers $\varepsilon_k^{(n)}$ can be displayed in a double entry table, called *the ε-array*, where the index k denotes the k-th column and the superscript n determines the n-th descending diagonal. The rule (2.12) relates numbers located at the four vertices of a rhombus in Table 2.1.

$$\begin{array}{ccc} & \varepsilon_k^{(n)} & \\ \varepsilon_{k-1}^{(n+1)} & & \varepsilon_{k+1}^{(n)} \\ & \varepsilon_k^{(n+1)} & \end{array}$$

Thus, knowing $\varepsilon_{k-1}^{(n+1)}$, $\varepsilon_k^{(n)}$, and $\varepsilon_k^{(n+1)}$ it is possible to compute $\varepsilon_{k+1}^{(n)}$ as

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

Instead of proceeding by columns, Wynn proposed a technique of computation by ascending diagonals. Given a new element S_n, the rule of the ε-algorithm allows us to proceed in the ε-array from top to bottom in order to complete its ascendant diagonal from left to right using only the preceding ascendant diagonal.

The most important property of this algorithm is the following relationship, proved by Wynn that links his algorithm to the Shanks' transformation:

$$e_k(S_n) = \varepsilon_{2k}^{(n)}$$

This important property is hiding a problematic aspect that we have mentioned at the end of the previous section. If the algorithm works well, that is we have a good acceleration in a column k at could happens that $\varepsilon_{2k}^{(n)}$ and $\varepsilon_{2k}^{(n+1)}$ are "almost" equal and, thus, in the computation of $\varepsilon_{2k+1}^{(n)}$ there is an important cancellation error that will occur in the difference $\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}$. Thus $\varepsilon_{2k+1}^{(n)}$ will be large and badly computed and since it appears in the denominator of the rule for $\varepsilon_{2k+2}^{(n)}$ it will cause numerical instability in the next even column. In order to avoid this kind of computational drawbacks Wynn introduced particular rules for the ε-algorithm. These rules involve more terms then the rhombus described above therefore require a more careful implementation.

In this thesis which wants to improve the performance of extrapolation by increasing the parameter k, we need to manage the so-called singularities. We have an **isolated singularity** when there are only two consecutive elements, $\varepsilon_k^{(n+1)}$ and $\varepsilon_k^{(n+1)}$, in a column on ε-array that is too close, i.e. fixed a parameter $p \in \mathbb{N}$ when

$$\frac{|\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}|}{|\varepsilon_k^{(n)}|} < 10^{-p}$$

When there are three or more elements involved in the singularity, it's called **non isolated singularity**. The name probably derives from the article [30] published by Wynn where he describes a first particular rule to fix only the singularities that he calls 'isolated' and he distinguishes them from the other cases.
The rule proposed by Wynn is able to treat only the isolated singularities and it is implemented in the Matlab toolbox `EPSfun` developed by Brezinski and Redivo-Zaglia [8].

In this work we present directly the Cordellier's particular rule that extends the Wynn's particular rules in order to study the singularity not isolated where the involved elements are three or more. We use the Table 2.2 to present a singularity of size $m+1$ and to give the notation that we use to describe the rule. The area affected by the singularity lies inside the square block, which we will call *singular square block* of size $m+1$. In the Cordellier's article [14] the quantities N_i, S_i, W_i, E_i showed in the Table 2.2, are related by the following relation:

$$(E_i - C)^{-1} + (W_i - C)^{-1} = (S_i - C)^{-1} + (N_i - C)^{-1}, \quad i = 1, \ldots, m \qquad (2.13)$$

$\varepsilon_{k-1}^{(n)} = N_0$		$\varepsilon_{k+1}^{(n-1)} = N_1$		$\cdots$	$\varepsilon_{k+2m-1}^{(n-m)} = N_m$		$\varepsilon_{k+2m+1}^{(n-m-1)} = N_{m+1}$
	$\varepsilon_k^{(n)} \sim \alpha$		$\cdots$	$\cdots$		$\varepsilon_{k+2m}^{(n-m)} \sim \alpha$	
$\varepsilon_{k-1}^{(n+1)} = W_1$		$\varepsilon_{k+1}^{(n)} \sim C$		$\cdots$	$\varepsilon_{k+2m-1}^{(n-m+1)} \sim C$		$\varepsilon_{k+2m+1}^{(n-m)} = E_m$
	$\varepsilon_k^{(n+1)} \sim \alpha$		ω	$\cdots$ ω		$\varepsilon_{k+2m}^{(n-m+1)} \sim \alpha$	
$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$\vdots$		ω	$\cdots$ ω		$\cdots$	
$\varepsilon_{k-1}^{(n+m)} = W_m$		$\varepsilon_{k+1}^{(n+m-1)} \sim C$		$\cdots$	$\varepsilon_{k+2m-1}^{(n)} \sim C$		$\varepsilon_{k+2m+1}^{(n-1)} = E_1$
	$\varepsilon_k^{(n+m)} \sim \alpha$		$\cdots$	$\cdots$		$\varepsilon_{k+2m}^{(n)} \sim \alpha$	
$\varepsilon_{k-1}^{(n+m+1)} = S_{m+1}$		$\varepsilon_{k+1}^{(n+m)} = S_m$		$\cdots$	$\varepsilon_{k+2m-1}^{(n+1)} = S_1$		$\varepsilon_{k+2m+1}^{(n)} = S_0$

Table 2.2: Part of an ε-array with a singularity of size $m+1$.

From the preceding relation we can compute the E_i using the following rule:

$$\begin{aligned} E_i &= r_i \left(1 + \frac{r_i}{C}\right)^{-1} \\ r_i &= S_i \left(1 - \frac{S_i}{C}\right)^{-1} + N_i \left(1 - \frac{N_i}{C}\right)^{-1} - W_i \left(1 - \frac{W_i}{C}\right)^{-1} \end{aligned} \tag{2.14}$$

The Cordellier's particular rule can also be extended to situations where there is no singularity and where the singularities don't be caused by exactly equal values. It's sufficient to replace the C parameter in the expression (2.13) with the exact value $\varepsilon_k^{(n+i)}$ where it's finite, otherwise we impose the value ∞. For a singular square block of size $m+1$ showed in Table 2.2, the quantity E_i which corresponds to the element $\varepsilon_{k+2(m+1)-1}^{(n-i)}$ is computed by the formula

$$\begin{aligned} E_i &= r_i \left(1 + \frac{r_i}{C_i}\right)^{-1} \\ r_i &= S_i \left(1 - \frac{S_i}{C_i}\right)^{-1} + N_i \left(1 - \frac{N_i}{C_i}\right)^{-1} - W_i \left(1 - \frac{W_i}{C_i}\right)^{-1} \end{aligned}$$

with $S_i = \varepsilon_{k+2(m+1-i)-1}^{(n+i)}$, $N_i = \varepsilon_{k+2i-1}^{(n-i)}$, $W_i = \varepsilon_{k-1}^{(n+i)}$, $C_i = \varepsilon_{k+1}^{(n+i-1)}$.

In order to handle non isolated singularities that could be created during extrapolation, we have implemented the `handle MATLAB class SEA` (Scalar Epsilon Algorithm) and the `handle MATLAB class SEA_sing_class`. The latter is useful to store the quantities needed for computing the Cordellier's particular rule by ascending diagonals. The MATLAB code is shown in the Appendix C of this thesis. We refer to the Karapiperi's thesis [21] for numerical example because applying the same sequence we obtain the same results thanks to our implementation.

2.3 The simplified topological ε-algorithm

In the precedent section we presented the Shanks' transformation for a scalar sequence but our problem is to accelerate the convergence of the vector sequence $\mathbf{u}^{(n)}$ given by (2.1). The first work about acceleration of the vector sequences is the article [31] published by Wynn in 1962. It was obtained directly as a generalization of his scalar algorithm without algebraic foundation. In 1991 Brezinski [7] proposed a different way to obtain a vector generalization of Shanks' transformations called the *topological Shanks' transformations*. In this section we want to present the **simplified topological ε-algorithm** (STEA) [11] [12] that introduces new algorithms to implement the topological Shanks' transformations. These new algorithms work better than the original Brezinski's implementation from a theoretical point of view and for numerical reasons.

Let $(\mathbf{S}_n)$ be a sequence of elements of a vector space E on a field $\mathbb{K}$ ($\mathbb{R}$ or $\mathbb{C}$). The procedure follows the spirit of Shanks' for the scalar ε-algorithm that we see in §2.1. We start from a relation similar to (2.6) that represents the implicit form of the kernel,

$$a_0(\mathbf{S}_n - \mathbf{S}) + \cdots + a_k(\mathbf{S}_{n+k} - \mathbf{S}) = \mathbf{0} \in E \tag{2.15}$$

where now $\mathbf{S}_n$, $\mathbf{S} \in E$ and a_i are scalars with $a_0\, a_k \neq 0$ and $a_0 + \cdots + a_k \neq 0$. Adding the condition $a_0 + \cdots + a_k = 1$ which does not restrict the generality, we have the following expression to compute $\mathbf{S}$ similar to (2.7)

$$\mathbf{S} = a_0\, \mathbf{S}_n + \cdots + a_k\, \mathbf{S}_{n+k}, \quad n = 0, 1, \ldots \tag{2.16}$$

Hence we have to determinate the coefficients a_i from (2.15) and we eliminate the dependence from the term $\mathbf{S}$ taking (2.16) for n and $n+1$, subtracting the two relations and writing

$$a_0\, \Delta\mathbf{S}_n + \cdots + a_k\, \Delta\mathbf{S}_{n+k} = \mathbf{0}, \quad n = 0, 1, \ldots \tag{2.17}$$

Now let $\mathbf{y}$ be an arbitrary vector in the algebraic dual space of E, E^*. We take the duality product of the relations (2.17) with $\mathbf{y}$ for $i = n, \ldots, n+k-1$ and we obtain the following k equations:

$$a_0 < \mathbf{y}, \Delta\mathbf{S}_i > + \cdots + a_k < \mathbf{y}, \Delta\mathbf{S}_{i+k} >= 0, \quad i = n, \cdots, n+k-1 \tag{2.18}$$

As we made for the scalar sequences, the coefficients can be computed by solving a linear system. If we take a $(\mathbf{S}_n)$ which doesn't belong to the kernel (2.15) and we impose for a dual vector $\mathbf{y}$ the system (2.18) and the normal condition: we may solve the system

$$\begin{cases} a_0^{(n,k)} + \ldots + a_k^{(n,k)} = 1 \\ a_0^{(n,k)} < \mathbf{y}, \Delta\mathbf{S}_n > + \ldots + a_k^{(n,k)} < \mathbf{y}, \Delta\mathbf{S}_{n+k} >= 0 \\ \vdots \\ a_0^{(n,k)} < \mathbf{y}, \Delta\mathbf{S}_n > + \ldots + a_k^{(n,k)} < \mathbf{y}, \Delta\mathbf{S}_{n+k} >= 0 \end{cases} \tag{2.19}$$

where now the coefficients depend on n and k (denoted by the superscript (n,k)). From the system (2.19), whose determinant is assumed to be nonzero, we compute $a_i^{(n,k)}$ and then we have the transformation defined by

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

for any i. When $i = 0$ we have the first topological Shanks' transformation (STEA1), for $i = k$ we have the second topological Shanks' transformation (STEA2). The recently experiments (see [11]) show that the second topological Shanks' transformation, from the numerical point of view, is better than the first one. We denote the STEA2 with a $\sim$ symbol and we have:

$$\tilde{\mathbf{e}}_k(\mathbf{S}_n) = a_0^{(n,k)}\,\mathbf{S}_{n+k} + \cdots + a_k^{(n,k)}\,\mathbf{S}_{n+2k} \quad n,k = 0,1,\ldots \tag{2.20}$$

As in the scalar case, $\tilde{\mathbf{e}}_k(\mathbf{S}_n)$ can be expressed as a ratio of two determinants:

$$\tilde{\mathbf{e}}_k(\mathbf{S}_n) = \frac{\begin{vmatrix} \mathbf{S}_{n+k} & \cdots & \mathbf{S}_{n+2k} \\ <\mathbf{y},\Delta\mathbf{S}_n> & \cdots & <\mathbf{y},\Delta\mathbf{S}_{n+k}> \\ \vdots & & \vdots \\ <\mathbf{y},\Delta\mathbf{S}_{n+k-1}> & \cdots & <\mathbf{y},\Delta\mathbf{S}_{n+2k-1}> \end{vmatrix}}{\begin{vmatrix} 1 & \cdots & 1 \\ <\mathbf{y},\Delta\mathbf{S}_n> & \cdots & <\mathbf{y},\Delta\mathbf{S}_{n+k}> \\ \vdots & & \vdots \\ <\mathbf{y},\Delta\mathbf{S}_{n+k-1}> & \cdots & <\mathbf{y},\Delta\mathbf{S}_{n+2k-1}> \end{vmatrix}} \tag{2.21}$$

where the determinant in the numerator denotes the element of E obtained by developing it with respect to its first row by the classical rule for expanding a determinant.

In the article [7], Brezinski proposes his *topological ε-algorithm* TEA2. He introduces like for ε-algorithm the quantities $\widetilde{\boldsymbol{\varepsilon}}_k^{(n)}$ given by the following rules

$$\begin{cases} \widetilde{\boldsymbol{\varepsilon}}_{-1}^{(n)} = \mathbf{0} \in E^*, & n = 0,1,\ldots \\ \widetilde{\boldsymbol{\varepsilon}}(n)_0 = \mathbf{S}_n \in E, & n = 0,1,\ldots \\ \widetilde{\boldsymbol{\varepsilon}}_{2k+1}^{(n)} = \widetilde{\boldsymbol{\varepsilon}}_{2k-1}^{(n+1)} + \dfrac{\mathbf{y}}{<\mathbf{y},\widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} - \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n)}>} \in E^*, & k,n = 0,1,\ldots \\ \widetilde{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} = \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \dfrac{\widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}}{<\widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}, \widetilde{\boldsymbol{\varepsilon}}_{2k+1}^{(n+1)} - \widetilde{\boldsymbol{\varepsilon}}_{2k+1}^{(n)}>} \in E, & k,n = 0,1,\ldots \end{cases}$$

As in the scalar case, the topological ε-algorithm is connected to the topological Shanks' transformation via the following relation:

$$\tilde{\mathbf{e}}_k(\mathbf{S}_n) = \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n)}$$

But from the statement above it is evident that the duality product that appears in the rules is difficult to be handled on a computer, apart when $(\mathbf{S}_n)$ are real or complex vectors and the duality product is the usual inner product between vectors.

In [11], Brezinski and Redivo-Zaglia highlight the fact that this algorithm can be greatly simplified thank to some relations existing when we apply the scalar ε-algorithm to $(S_n) = (< \mathbf{y}, \mathbf{S}_n >)$. They exploit this property in order to obtain STEA2. Its rules becomes:

$$\begin{cases} \widehat{\boldsymbol{\varepsilon}}_0^{(n)} = \mathbf{S}_n \in E, & n = 0, 1, \ldots \\ \widehat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} = \widehat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \dfrac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}} \left(\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \widehat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} \right) \in E, & k, n = 0, 1, \ldots \end{cases} \tag{2.22}$$

where the $\varepsilon_k^{(n)}$ are obtained by using the scalar ε-algorithm applied to the scalar sequence $(S_n) := (< \mathbf{y}, \mathbf{S}_n >)$. For a more detailed description on the derivation of the aforementioned formulas refer to Brezinski, Redivo-Zaglia's articles [11][12] where several convergence and acceleration results for STEA are also given and where the Matlab toolbox `EPSfun` is described. For instance, for the class of *totally monotonic sequence* the authors give interesting theoretical results. During our investigation we try to use this results in order to justify the convergence of our propose. But we couldn't prove that our vector sequence $\mathbf{u}^{(n)}$ given by (2.1) is a totally monotonic sequence.

Leaving aside the aspects of convergence, from a numerical point of view, the simplified topological ε-algorithm is better than the topological ε-algorithm, since it helps us to overcome several computational problems. Furthermore we can use the particular rules shown in precedent section for the scalar ε-algorithm, in order to handle the potential singularities during the computation of the scalars $\varepsilon_k^{(n)}$ and thus improve the overall numerical stability. The STEA requires the storage of less elements because the relation given by (2.22) uses only the even elements of the last ascending diagonal of the vector ε-array. However a questions remain outstanding in the literature: what dual element $\mathbf{y}$ holds the most performance in the STEA2? We try to answer through experiments in the next chapter because a theoretical treatment of the question is not yet know. In order to compare with the Brezinski, Redivo-Zaglia's implementation we use for the example which will appear in this chapter, the vector $\mathbf{y} \equiv 1$ whose components are all 1.

2.4 Numerical results with fixed step size

In this section we want to present the performance of the extrapolation method based on simplified topological ε-algorithm as it has been proposed in [10], for solving by using a fixed step size $\alpha^{(n)} = \alpha \, \forall n$. In the next chapter we present other numerical examples but in this chapter we take this opportunity to focus our attention on Example 2 and on Example 3 introduced and discussed in the section §1.4. As previously said, we will maintain the same examples' number of article [10] because it will be natural to make comparisons in order to affirm the goodness of our work. But we must take into account that Brezinski and Redivo-Zaglia use an inefficient quadrature formula. Therefore in the following pages we want to re-propose the phenomenon highlighted in the use of a trapezoidal instead of the Gaussian formula, as in section §1.4 after the introduction of the acceleration.

In their article, Brezinski and Redivo-Zaglia compare the performance between *acceleration method* and *restarted method*. The restarted method is basically an acceleration method where after a fixed number of iterations the method is re-initialized using as initial vector the last extrapolated vector. This method is applicable only in problems where we want to determine the fixed point of an operator, like in this case. It is intuitive that restarted method is not exploiting all the information accumulated in all precedent iterations. In their conclusion [10, p. 25], the authors prefer it to acceleration method because in their results the restarted method seems to be more effective. We want to prove through a practical point of view that acceleration method exploited in all its potential is very efficient, and the restarted method is therefore unnecessary in this context.

A problem which appears in their article where the use of the second method seems indispensable is exactly Example 2. As aforementioned we have the following integral equation

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

We had more than an opportunity to discuss some aspects of Example 2 in Chapter 1. We will deal with the theoretical aspects highlighted in detail to give a complete description of what happens from a numerical point of view. We are sure about the convergence of the original sequence only if we stay within the confidential region described by the conditions of sufficient Theorem 1.3. Outside of it we have no theoretical results to ensure the convergence of the original sequence and this partly explains the reason why in the article [10] the acceleration method does not converge. Furthermore the Theorem 1.3 gives us some sufficient conditions to ensure the convergence on solution $u_2^*(t) = -\frac{1}{5}\cos\left(\frac{\pi\, t}{4}\right)$; on the contrary, in [10] the authors show that their method converges on the $u_1^*(t) = -\frac{1}{5}\cos\left(\frac{\pi\, t}{4}\right)$ solution where even the Lipschitz's condition that they had

supposed is not valid (see Observation 1.1 for details). We want to show what happens if we act "outside" the confidential region, in particular we will represent the situation created in their article and that has determined their deduction. We prove in Observation 1.2 that the region where the sufficient condition are satisfied is

$$\widetilde{M} = \left\{ \mathbf{v} \in \mathbb{R}^{p+1} \text{ such that } \|\mathbf{v}\|_\infty < \frac{4}{3\sqrt{2\pi}} \right\}$$

In order to show in a Cartesian plan the $p+1$-vectors, we introduce the projection map $\psi : \mathbb{R}^{p+1} \to \mathbb{R}^2$ that associate to the vector $\mathbf{v} = (v_0, \ldots, v_p) \in \mathbb{R}^{p+1}$ with the minimum and maximum of coordinates, that is

$$\psi(\mathbf{v}) := \left(\min_{i=0,\ldots,p} v_i, \max_{i=0,\ldots,p} v_i \right).$$

We take the software implemented by Brezinski and Redivo-Zaglia downloaded from `MathWorks File Exchange #65048` and we plot the sequence of vectors obtained by STEA2[5] for different starting vectors. In the same figure we represent the sequence obtained by acceleration method, `AM - STEA2`, and the restarted method, `RM - STEA2`. In Cartesian plan, the region $\widetilde{M}$ corresponds to the triangle of vertices: $\left(-\frac{4}{3\sqrt{2\pi}}, -\frac{4}{3\sqrt{2\pi}}\right)$, $\left(\frac{4}{3\sqrt{2\pi}}, \frac{4}{3\sqrt{2\pi}}\right)$, and $\left(-\frac{4}{3\sqrt{2\pi}}, \frac{4}{3\sqrt{2\pi}}\right)$.

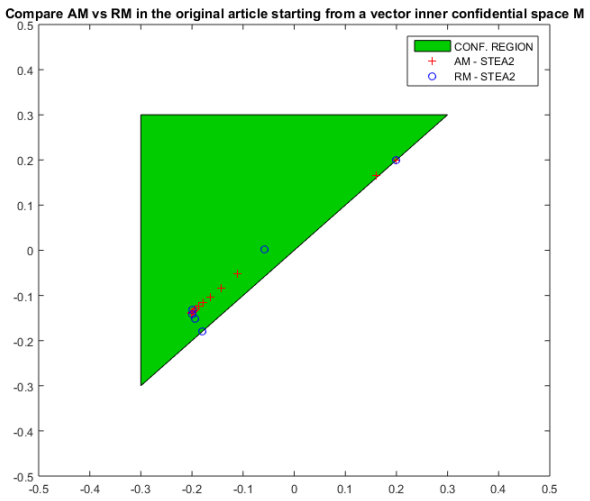

Figure 2.1: Example 2 - Representation of the convergence of the method [10] for a starting vector inner confidential region in the Cartesian plane.

[5]For those experiments we use the parameter used by Brezinski and Redivo-Zaglia: $\alpha = 0.1$, $\mathbf{y} \equiv 1$, $2k = 6$, Max_Iter= 50, Tol= 10^{-8}, $M = 20$ and using trapezoidal rule with $p = 21$. The parameter M is used in [10] as stopping criterion because there is no advantage in continuing the iterations when an oscillation arise due to the instability around the machine's precision (see [10, p.16] for details).

The first starting vector that we consider is $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$. It belongs to the confidential region $\widetilde{M}$ therefore we expect the convergence to the Cartesian's point $\psi(\mathbf{u}_2^*) \sim (-0.13, -0.21)$, individuated by the solution $u_2^*(t) = -\frac{1}{5}\cos\left(\frac{\pi\, t}{4}\right)$. In fact this is exactly what we observe in the Figure 2.1.

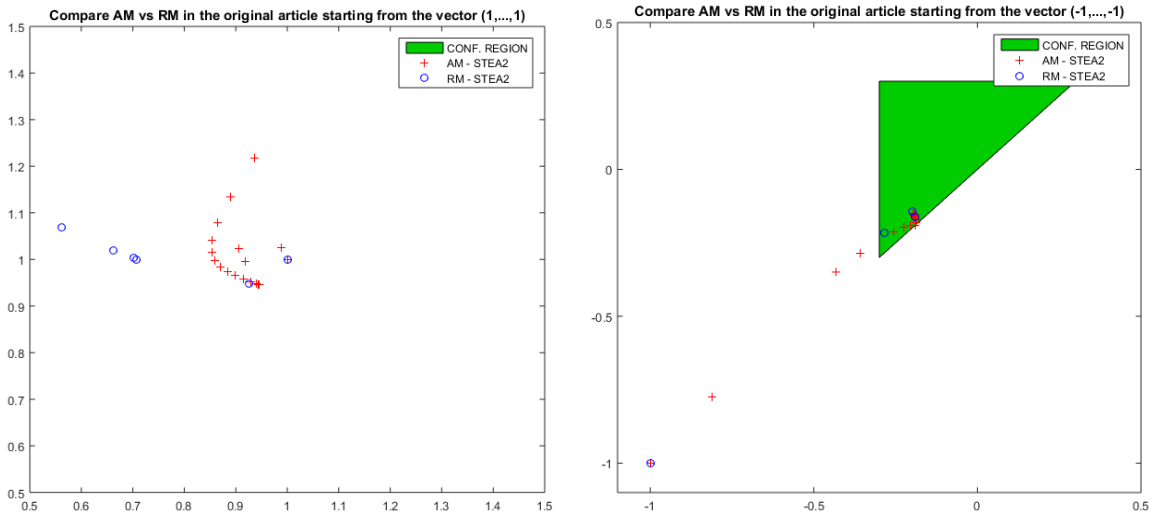

Figure 2.2: Example 2 - Representation in the Cartesian plane of the convergence of the [10] method for two different starting vectors outside confidential region. In the left figure we show that AM-STEA2 method doesn't converge if starting from vector $\mathbf{u}^{(0)} \equiv 1$ and we can see that at every iteration the sequence moves away from the Cartesian's point $\psi(\mathbf{u}_1^*)$. While in the right figure we can observe that we reach the confidential region starting from $\mathbf{u}^{(0)} \equiv -1$ and therefore converges to the Cartesian's point $\psi(\mathbf{u}_2^*)$.

The second starting vector that we consider is $\mathbf{u}^{(0)} \equiv 1$ which corresponds to the situation described on article [10]. As the authors themselves have noticed, on the left of Figure 2.2 we see that the restarted method converge on $\psi(\tilde{\mathbf{u}}_1) \sim (-0.98, -0.98)$, while the acceleration method do not converge. Then it might be thought that any element outside the confidential region does not converge or converge to the solution $u_1^*(t) = \cos\left(\frac{\pi\, t}{4}\right)$. However as we can see on the right of Figure 2.2 this is not true: we represent the convergence for a starting vector $\mathbf{u}^{(0)} \equiv -1$ which stands outside the confidential region $\widetilde{M}$. The sequences obtained by both methods converge into the confidential region therefore they converge to the solution u_2^*.

We do not think it's necessary to show additional graphs and examples[6], since what should remain of this discussion is the importance of the confidential region.

[6]We can also obtain the case where both methods do not converge. This happens for example when we take as starting vector $\mathbf{u}^{(0)} = (-1, -1, -1, -1, -1, -1, -1, 1, 1, \ldots, 1)$.

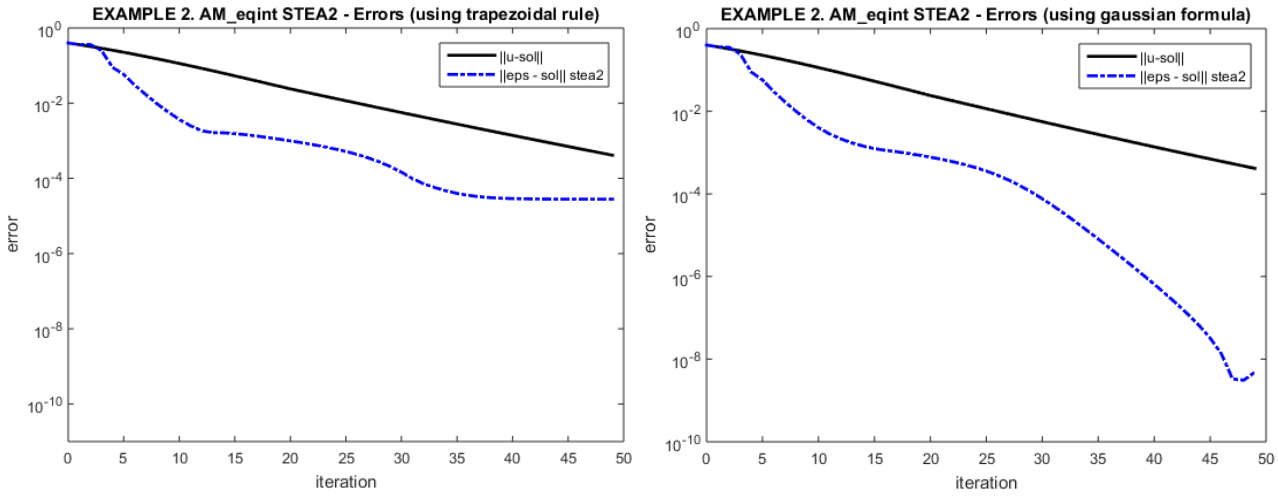

Figure 2.3: Example 2 - Comparison of AM_eqint with fixed $\alpha = 0.1$. On the left we report the errors using the trapezoidal rule with $p = 21$ and on the right the errors determined by the Gaussian formula with $p = 8$.

If we operate within it, we are sure that the algorithms that we will present in this work converge. Outside it or in cases where these conditions are never verified, we cannot be sure of converging to some element. In many practical cases, as in Example 3, we have already observed that RPIM leads to convergence, but we have not found theoretical results explaining this fact.

For the reasons expressed above we consider $u_2^*(t) = -\frac{1}{5}\cos\left(\frac{\pi t}{4}\right)$ as reference solution and the confidential region established by the interval $\widetilde{M} = \left(-\frac{4}{3\sqrt{2\pi}}, \frac{4}{3\sqrt{2\pi}}\right)$. In order to make a reasonable comparison we consider the same parameter[7] as Brezinski, Redivo-Zaglia's experiment [10, p. 19], and to hold into account the influence of the quadrature formula in the errors more than in the residual we show in Figure 2.3 a comparison by using the trapezoidal rule with 22 nodes (left) and the Gaussian formula (right). The Figure 2.3 shows again the importance of the choice of the quadrature formula but we wonder if this is really the maximum performance attainable with this algorithm. Brezinski and Redivo-Zaglia indicate in their article [10, p. 15] that the parameter α have to be chosen in order to obtain the convergence of the iterations and for this reason they take small α values (like 0.3,0.1). Instead in Chapter 1 we proved that the original sequence given by simple Picard iteration method (i.e. RPIM with $\alpha = 1$) converges to the solution within confidential region. In the Figure 2.4 we compare the errors determinate by the algorithm AM_eqint between $\alpha = 0.5$

[7]For those experiments with our implementation we initialize the parameter used in [10]: $\alpha = 0.1$, $\mathbf{y} \equiv 1$, $2k = 4$, Max_Iter= 50, Tol= 10^{-8} and starting from $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$.

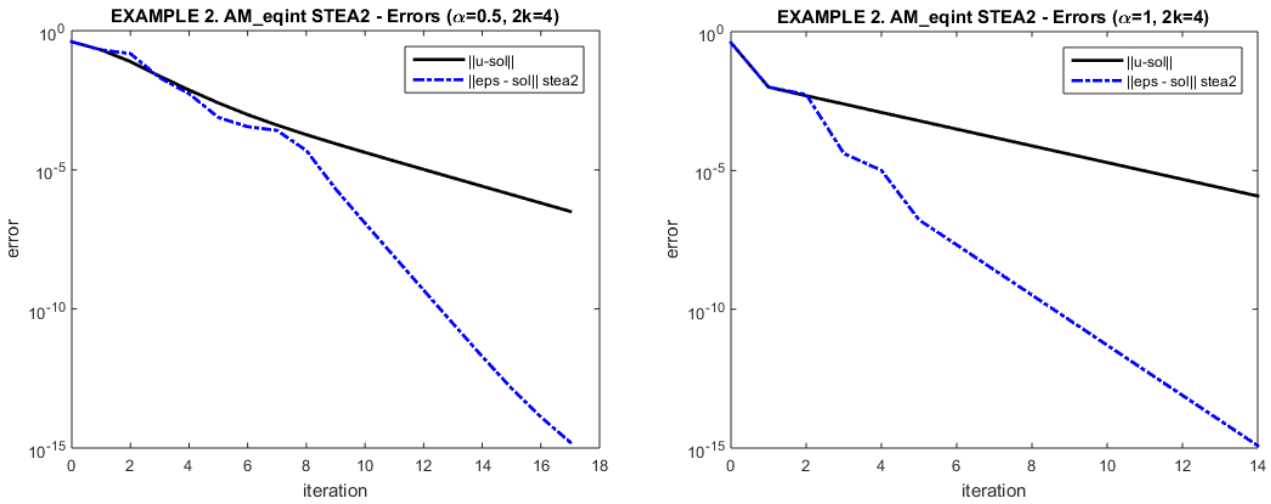

Figure 2.4: Example 2 - Comparison of AM_eqint between $\alpha = 0.5$ and $\alpha = 1$. Both experiments are determined with parameter $2k = 4$.

and $\alpha = 1$ with $2k = 4$. We note that $\alpha = 1$ determinates a better performance than $\alpha = 0.5$ because in the first one the extrapolated sequence converges to the machine precision in only 14 iteration[8]. We have observed the same phenomenon during our experiments for different cases where the conditions of Theorem 1.3 are verified. Therefore for these cases we will often keep frequently $\alpha = 1$ for a fixed step size strategy in order to obtain a higher convergence.

In above experiments we have kept the order of Shanks' transformation to $2k = 4$ as in [10]. But given the promising behaviour of the original sequence, we want to determine an improvement in the performance of acceleration algorithm through the increase of this parameter. In the Table 2.3 we compare the performance of AM_eqint between $2k = 4$ (the results that we show in the Figure 2.4) and $2k = 16$, for $\alpha = 1$ and $\alpha = 0.5$. We note that for $\alpha = 1$ the increase to $2k = 16$ allows to converge to the machine precision in only 10 iterations instead of 14 iterations. But for $\alpha = 0.5$ this is not true because we don't notice any net improvement in performance. This aspect suggests that in general taking a value of $2k$ high in order to exploit all the information so far acquired, iteration after iteration, may not be effective. In the next section we will have the opportunity to discuss this aspect and understand how to adopt an adaptive strategy for step size in order to obtain an acceleration method with a non-irregular descent. What we would like it to remain about the fixed step size strategy is that in the experimental phase, within the confidence region, it will be convenient to use the simple Picard iteration but it will be advisable to test different values for the $2k$ parameter in order to determine what offers a better convergence.

[8]The value of errors and residuals for each iteration can be consulted on the Table 2.3.

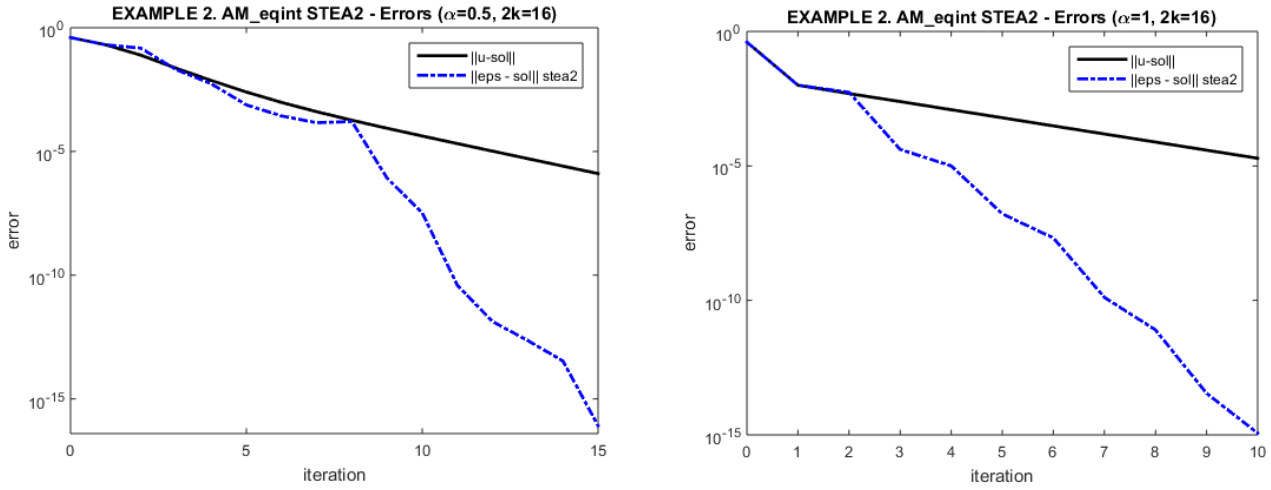

Figure 2.5: Example 2 - Comparison of AM_eqint with $2k = 16$ between fixed $\alpha = 0.5$ and $\alpha \equiv 1$. We increase the number of iterations involved in extrapolation method respect on the Figure 2.4 where we set $2k = 4$.

	α=0,5				**α = 1**			
	2k = 4		**2k = 16**		**2k = 4**		**2k = 16**	
n	**Residual**	**Error**	**Residual**	**Error**	**Residual**	**Error**	**Residual**	**Error**
0	3,90E-01	4,00E-01	3,90E-01	4,00E-01	3,90E-01	4,00E-01	3,90E-01	4,00E-01
1	2,55E-01	2,05E-01	2,55E-01	2,05E-01	1,49E-02	1,00E-02	1,49E-02	1,00E-02
2	2,56E-01	1,50E-01	2,56E-01	1,50E-01	8,20E-03	5,44E-03	8,20E-03	5,44E-03
3	3,08E-02	2,03E-02	3,08E-02	2,03E-02	6,17E-05	4,11E-05	6,17E-05	4,11E-05
4	7,58E-03	5,45E-03	7,58E-03	5,45E-03	1,51E-05	1,01E-05	1,51E-05	1,01E-05
5	7,92E-04	7,61E-04	7,92E-04	7,61E-04	2,48E-07	1,65E-07	2,48E-07	1,65E-07
6	3,37E-04	3,55E-04	2,59E-04	2,73E-04	3,14E-08	2,10E-08	3,14E-08	2,10E-08
7	2,47E-04	2,61E-04	1,37E-04	1,46E-04	3,90E-09	2,60E-09	1,95E-10	1,30E-10
8	4,69E-05	4,95E-05	1,53E-04	1,62E-04	4,89E-10	3,26E-10	1,20E-11	8,01E-12
9	1,93E-06	2,03E-06	7,96E-07	8,43E-07	6,11E-11	4,07E-11	5,27E-14	3,52E-14
10	1,15E-07	1,22E-07	3,12E-08	3,30E-08	7,64E-12	5,09E-12	1,78E-15	1,17E-15
11	7,17E-09	7,56E-09	3,71E-11	3,91E-11	9,55E-13	6,36E-13		
12	4,49E-10	4,74E-10	1,23E-12	1,33E-12	1,19E-13	7,95E-14		
13	2,84E-11	2,99E-11	2,10E-13	2,30E-13	1,48E-14	9,96E-15		
14	1,84E-12	1,94E-12	3,13E-14	3,42E-14	1,89E-15	1,22E-15		
15	1,31E-13	1,39E-13	1,39E-16	8,33E-17				
16	1,22E-14	1,30E-14						
17	1,50E-15	1,61E-15						

Table 2.3: Example 2 - Performance of fixed step size strategy for different values of α and $2k$ with $\mathbf{y} \equiv 1$, Max_iter= 50, Tol= $5 \cdot 10^{-15}$, using the Gaussian formula with $p = 8$, and starting from $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$.

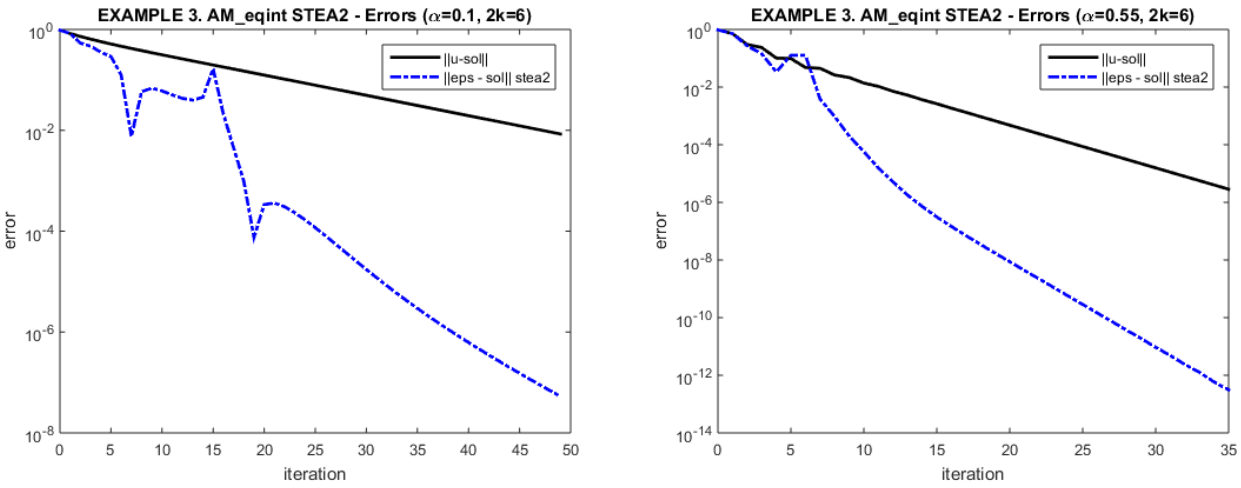

Figure 2.6: Example 3 - Comparison of AM_eqint for $2k = 6$ between $\alpha = 0.1$ (used in the article [10]) and $\alpha = 0.5$.

The fixed step size strategy is also useful in the study of integral equations that not satisfy the sufficient condition given by Theorem 1.3. Let us consider again the earlier Example 3 studied in §1.4 where we consider the integral equation with the following form:

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

We consider as reference solution $u^*(t) = t$ and choose the Gaussian formula with $p = 5$. In fact as we seen before, an inefficient quadrature formula makes it completely unnecessary to continue below the order of magnitude established by the quantity $\mathcal{E}^{\mathrm{QUAD}}(u^*)$. Using the same method used for Example 2 in §1.4, we can determinate that $\mathcal{E}^{\mathrm{QUAD}}(u^*) \sim 5 \cdot 10^{-4}$ and thus justify that the error in Figure 5 of [10, p. 21] stagnates to this order of magnitude despite the residual continues to descend. In Figure 2.6 (left) we see what the authors would have achieved by using the Gaussian formula with $p = 5$ instead of the trapezoidal rule with $p = 100$.

In view of the experience of the previous example we intend to increase α taking into account that the original sequence does not converge if $\alpha > 0.55$: hence we increase α to the upper limit $\alpha = 0.55$. In Figure 2.6 we compare the errors between $\alpha = 0.1$ (left) and $\alpha = 0.55$ (right) while keeping the other parameters[9] fixed. Both have a similar behaviour but in the first case $\alpha = 0.1$ is too small to provide an interesting performance even if it enjoys a very smooth descent

[9]For those experiments with our implementation we use the parameter: $\mathbf{y} \equiv 1$, $2k = 6$, Max_Iter= 50, Tol= $5 \cdot 10^{-15}$, using Gaussian formula with $p = 5$ and starting from $\mathbf{u}^{(0)} \equiv 1$.

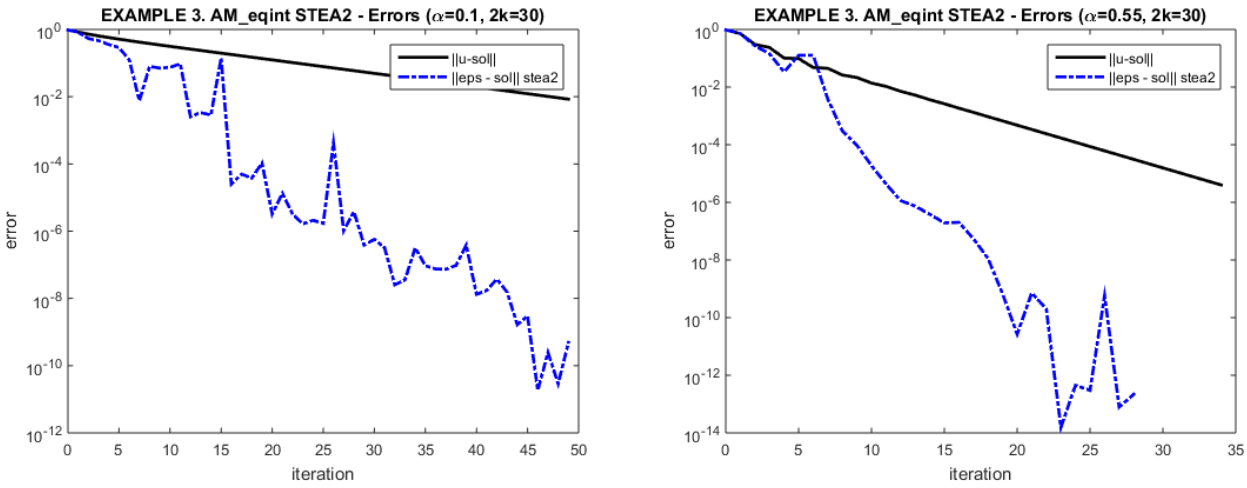

Figure 2.7: Example 3 - Comparison of AM_eqint for $2k = 30$ with fixed $\alpha = 0.1$ (using in the article [10]) and $\alpha = 0.5$.

from iteration 25. While in the second case we have a fairly fast descent starting from the seventh iteration. As we did in the previous example we increase the parameter $2k = 30$ in order to experience greater performance using all precedent iterations[10]. In Figure 2.7 we can see that in the first case the error goes down in an irregular way while in the second case we note up to iteration 23 a behaviour in which it already reaches a error of magnitude order 10^{-14} but after such iteration the error goes up. This phenomenon is recurrent when we act outside the region of confidence determined by the Theorem 1.3. For this reason the fixed step size strategy can be effective provided we pay attention to the choice of parameters α and $2k$ when we use this strategy outside the sufficient conditions of Theorem 1.3.

We report all results in the following Table 2.4.

[10]We had to stop the numerical extrapolation due to the presence of unmanageable non isolated singularities. These are caused by the cancellation errors that occur near the machine precision and they produce the `NaN` values.

	α=0,1				α = 0.55			
	2k = 6		2k = 30		2k = 6		2k = 30	
n	**Residual**	**Error**	**Residual**	**Error**	**Residual**	**Error**	**Residual**	**Error**
0	1,36E+00	9,66E-01	1,36E+00	9,66E-01	1,36E+00	9,66E-01	1,36E+00	0,966235
1	1,04E+00	8,30E-01	1,04E+00	8,30E-01	1,49E+00	7,14E-01	1,49E+00	0,713762
2	4,83E-01	5,37E-01	4,83E-01	5,37E-01	2,68E-01	2,71E-01	2,68E-01	0,270946
3	4,01E-01	4,65E-01	4,01E-01	4,65E-01	1,26E-01	1,43E-01	1,26E-01	0,142867
4	2,92E-01	3,56E-01	2,92E-01	3,56E-01	3,89E-02	3,35E-02	3,89E-02	0,033476
5	2,36E-01	2,90E-01	2,36E-01	2,90E-01	1,11E-01	1,27E-01	1,11E-01	0,127031
6	1,04E-01	1,25E-01	1,04E-01	1,25E-01	1,06E-01	1,27E-01	1,06E-01	0,127115
7	2,51E-02	7,55E-03	2,51E-02	7,55E-03	3,28E-03	3,83E-03	3,28E-03	0,003826
8	8,07E-02	5,82E-02	9,96E-02	7,89E-02	7,95E-04	9,55E-04	2,92E-04	0,000302
9	8,54E-02	6,87E-02	8,64E-02	7,08E-02	1,65E-04	1,99E-04	7,86E-05	9,57E-05
10	7,31E-02	6,05E-02	9,13E-02	7,47E-02	4,59E-05	5,74E-05	1,96E-05	1,89E-05
11	6,03E-02	5,04E-02	1,08E-01	9,49E-02	1,39E-05	1,55E-05	6,02E-06	4,32E-06
12	5,14E-02	4,30E-02	3,62E-03	2,55E-03	5,49E-06	5,14E-06	2,10E-06	1,15E-06
13	4,75E-02	3,98E-02	4,42E-03	3,36E-03	2,24E-06	1,78E-06	1,29E-06	7,38E-07
14	5,40E-02	4,52E-02	3,73E-03	2,85E-03	1,04E-06	7,20E-07	6,75E-07	3,83E-07
15	2,07E-01	1,67E-01	1,91E-01	1,40E-01	4,88E-07	3,10E-07	3,42E-07	1,94E-07
16	2,46E-02	2,13E-02	2,73E-05	2,52E-05	2,41E-07	1,45E-07	3,56E-07	2,02E-07
17	5,14E-03	4,53E-03	6,93E-05	4,99E-05	1,19E-07	6,98E-08	9,56E-08	5,43E-08
18	1,04E-03	9,89E-04	5,25E-05	3,79E-05	6,01E-08	3,46E-08	2,00E-08	1,14E-08
19	1,53E-04	7,50E-05	1,46E-04	1,06E-04	3,02E-08	1,72E-08	1,17E-09	6,66E-10
20	4,61E-04	3,36E-04	4,04E-06	3,22E-06	1,53E-08	8,68E-09	4,70E-11	2,67E-11
21	4,71E-04	3,61E-04	1,84E-05	1,34E-05	7,71E-09	4,38E-09	1,27E-09	7,20E-10
22	3,90E-04	3,04E-04	4,54E-06	3,17E-06	3,90E-09	2,21E-09	3,52E-10	2,00E-10
23	2,95E-04	2,32E-04	2,41E-06	1,65E-06	1,97E-09	1,12E-09	2,68E-14	1,55E-14
24	2,12E-04	1,68E-04	2,98E-06	2,10E-06	9,95E-10	5,65E-10	7,85E-13	4,47E-13
25	1,49E-04	1,18E-04	2,43E-06	1,72E-06	5,05E-10	2,86E-10	5,26E-13	2,99E-13
26	1,02E-04	8,09E-05	5,96E-04	4,22E-04	2,54E-10	1,44E-10	1,03E-09	5,82E-10
27	6,97E-05	5,51E-05	1,43E-06	1,03E-06	1,29E-10	7,34E-11	1,34E-13	7,87E-14
28	4,75E-05	3,75E-05	5,30E-06	3,80E-06	6,48E-11	3,68E-11	3,83E-13	2,19E-13
29	3,24E-05	2,55E-05	5,35E-07	3,86E-07	3,32E-11	1,89E-11		
30	2,22E-05	1,74E-05	7,91E-07	5,70E-07	1,65E-11	9,35E-12		
31	1,53E-05	1,20E-05	4,46E-07	3,22E-07	8,57E-12	4,86E-12		
32	1,06E-05	8,32E-06	3,63E-08	2,55E-08	4,18E-12	2,37E-12		
33	7,47E-06	5,83E-06	5,07E-08	3,59E-08	2,23E-12	1,27E-12		
34	5,30E-06	4,12E-06	4,11E-07	3,19E-07	1,03E-12	5,85E-13		
35	3,80E-06	2,94E-06	1,26E-07	9,37E-08	5,45E-13	3,10E-13		
36	2,74E-06	2,12E-06	1,04E-07	7,51E-08				
37	2,00E-06	1,54E-06	1,15E-07	7,38E-08				
38	1,47E-06	1,13E-06	1,19E-07	9,60E-08				
39	1,09E-06	8,36E-07	5,10E-07	3,85E-07				
40	8,09E-07	6,22E-07	1,73E-08	1,32E-08				
41	6,06E-07	4,65E-07	2,26E-08	1,74E-08				
42	4,56E-07	3,50E-07	5,02E-08	3,79E-08				
43	3,45E-07	2,64E-07	2,04E-08	1,52E-08				
44	2,61E-07	2,00E-07	2,16E-09	1,66E-09				
45	1,99E-07	1,53E-07	3,94E-09	3,01E-09				
46	1,52E-07	1,16E-07	3,50E-11	1,85E-11				
47	1,17E-07	8,95E-08	3,26E-10	2,46E-10				
48	8,98E-08	6,87E-08	3,74E-11	2,91E-11				
49	6,95E-08	5,31E-08	6,45E-10	4,88E-10				

Table 2.4: Example 3 - Performance of fixed step size strategy for different values of α and $2k$ with $\mathbf{y} \equiv 1$, Max_iter = 50, Tol = $5 \cdot 10^{-15}$, using the Gaussian formula with $p = 5$, and starting from $\mathbf{u}^{(0)} \equiv 1$.

2.5 A new adaptive strategy for step size

In the previous sections we have analysed the strategy used in [10] by Brezinski and Redivo-Zaglia and we have shown that choosing an appropriate quadrature formula and under appropriate conditions, this strategy produces good results. We will not be fobbed off with this performance; we want to understand how to improve it further. An aspect unexplored in the article [10] is the use of an adaptive strategy for the step size. In this section we want to understand what kind of strategy could facilitate acceleration. At first we will evaluate an intuitive but unsuccessful strategy and then we will present the adaptive strategy that we consider interesting from the practical point of view.

We recall the definition of the Relaxed Picard iteration method (RPIM) applied to our problem:

$$u_k^{(n+1)} = u_k^{(n)} + \alpha^{(n+1)} \left(f_k + \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i^{(n)}) - u_k^{(n)} \right) \quad k = 0, \ldots, p \tag{2.23}$$

In order to simplify the notation we introduce the following notation that explains the dependence on a generic α step size:

$$u_k^{(n+1);\,\alpha} = u_k^{(n)} + \alpha \left(f_k + \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i^{(n)}) - u_k^{(n)} \right) \tag{2.24}$$

A first idea that we intend to study is to determine the $\alpha^{(n+1)}$ at every iteration such that

$$\alpha^{(n+1)} := \underset{\widetilde{\alpha}>0}{\operatorname{argmin}} \left\| u_k^{(n+1);\,\widetilde{\alpha}} - f_k - \sum_{i=0}^{p} w_i \, \mathbf{K}_{k,i}(u_i^{(n+1);\,\widetilde{\alpha}}) \right\|_{\infty} . \tag{2.25}$$

In fact we have already proved in the Corollary 1.1 that the error is well approximated by the residual. Hence "reduce the residual" corresponds exactly to "reduce the error". Moreover, the experience of the last section pushes us in this direction given that a faster convergence of the original sequence corresponded to a faster extrapolated sequence[11].

To experience this idea and realize the performance we sample the domain and for this discretization we measure the residual given by (2.25) and we determine a good minimum approximation. At this level of evaluation it is not essential to understand whether we are getting exactly the minimum but the concept of reducing the residual. The one dimensional case, $\Omega = [a, b]$, is easy to implement using the `MATLAB`'s function `linspace(a,b,Num)` where `Num` is the number

[11] This is false in a general case as we will see in the following.

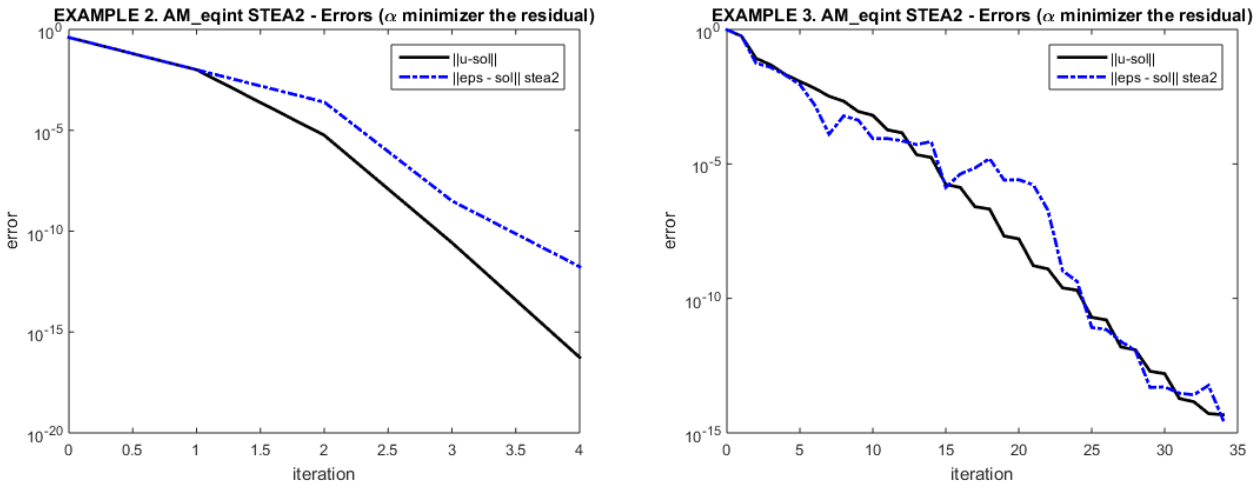

Figure 2.8: Comparison of AM_eqint with α "minimizer the residual" strategy for Example 2 (the left) and for Example 3 (the right).

of samples. In Figure 2.8 we propose[12] the behaviour of this method for Example 2 (with $2k = 4$) and for Example 3 (with $2k = 10$) where $\alpha^{(n)}$ is the minimizer of the residual. Of course the original sequence is very fast, but to get it we tested a thousand different α to each iterations and choose the best one. Even if this procedure is inefficient, the common optimization method cannot be applied to this case because the $\|\cdot\|_\infty$ norm is not derivable. Therefore, regardless of which method has been chosen, it's expensive and an acceleration method is essential. But in our case extrapolation gives unemployable results for both the examples. This result seems contradictory to what was said earlier but in reality, it is a well known phenomenon that concerns original sequences too fast to be accelerated by extrapolation method.

These results suggest that the effort needed to achieve a faster convergence of original sequence will probably not facilitate the acceleration based on extrapolation method. This leads us to evaluate an adaptive strategy in order to bring the sequence closer to the kernel of the transformation, $\mathcal{K}_T$. In fact as we have already anticipated in the Observation 2.1, it seems that the "closer" a sequence is to the kernel, the faster the transformed sequence will converge. For this reason we observe that a sequence $(\mathbf{u}^{(n)})$ for which exist $\lambda \in (-1, 1)$ such that

$$\frac{< \mathbf{y}, \mathbf{u}^{(n+1)} - \mathbf{u}^* >}{< \mathbf{y}, \mathbf{u}^{(n)} - \mathbf{u}^* >} = \lambda \tag{2.26}$$

stays in the kernel of scalar transformation associated to topological Shanks'

[12]For those experiments with our implementation we use the parameter: $\mathbf{y} \equiv 1$, Max_Iter = 50, and Tol = 10^{-15} and choosing the same quadrature formulas and the same starting vectors as before.

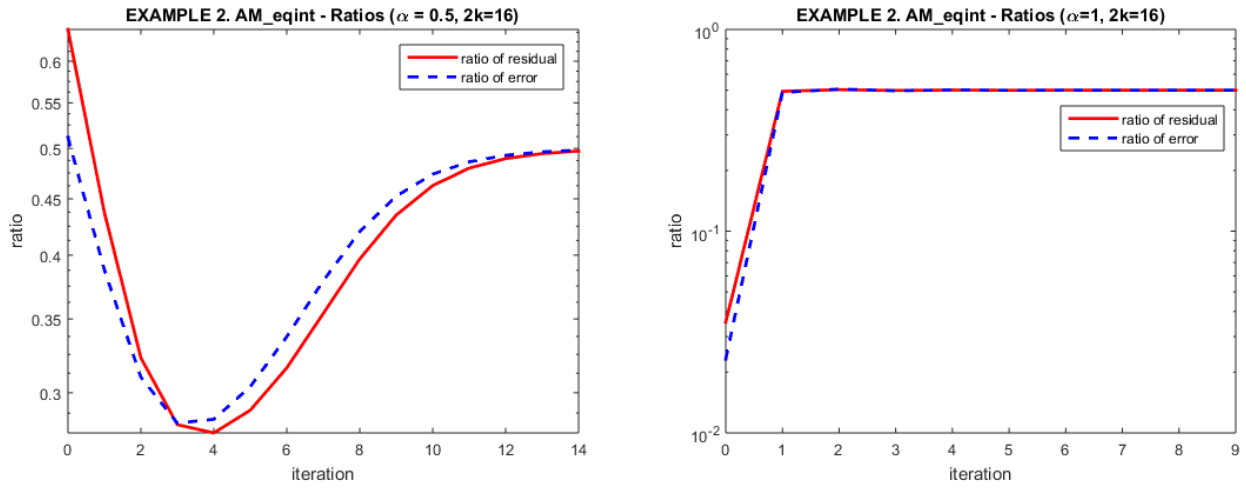

Figure 2.9: Example 2 - Comparison the ratios of AM_eqint with $2k = 16$ for fixed $\alpha = 0.5$ and for fixed $\alpha = 1$ showed in Figure 2.5.

transformation. In fact, we have seen in section §2.3 that this kernel is given by (2.18), i.e. the sequence $(\mathbf{u}^{(n)})$ for which exist $a_0, \ldots, a_k$

$$a_0 < \mathbf{y}, \mathbf{u}^{(n)} - \mathbf{u}^* > + \cdots + a_k < \mathbf{y}, \mathbf{u}^{(n+k)} - \mathbf{u}^* >= 0$$

where $a_0\, a_k \neq 0$ and $a_0 + \cdots + a_k \neq 0$. Hence for sequences of type (2.26) the above sequence become $\left(a_0 + \cdots + a_k\lambda^k\right) < \mathbf{y}, \mathbf{u}^{(n)} - \mathbf{u}^* >= 0$ and we can prove that this type of sequence belongs to the kernel of scalar transformation associated to topological Shanks' transformation using $a_0 = \frac{\lambda(1-\lambda^k)}{1-\lambda}$ and the other $a_i = 1$.
We would like to observe the behaviour of the ratio given by (2.26) and to which we referred as **ratio of errors** for the example that we studied in precedent section. We want to verify that the experiments in which an irregular descent occurred correspond to non-stationary behaviours and vice versa. We note that this ratio cannot be directly computed because a priori the exact solution is unknown. For this reason we introduce the **ratio of residuals** given by

$$\frac{< \mathbf{y}, \mathbf{u}^{(n+1)} - \mathbf{T}[\mathbf{u}^{(n+1)}] >}{< \mathbf{y}, \mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}] >} \tag{2.27}$$

We can observe in the following example that the behaviour of ratios is similar, nevertheless having attempted to obtain a theoretical proof that the ratios is related have been vain.

As we can see in Figure 2.5, for Example 2 we have obtained two different behaviours of extrapolated sequence. For fixed $\alpha = 0.5$ the error of the extrapolated sequence remains close to the error of the original sequence for the first 8 iterations after which it converges. While for $\alpha = 1$ the error drops steadily

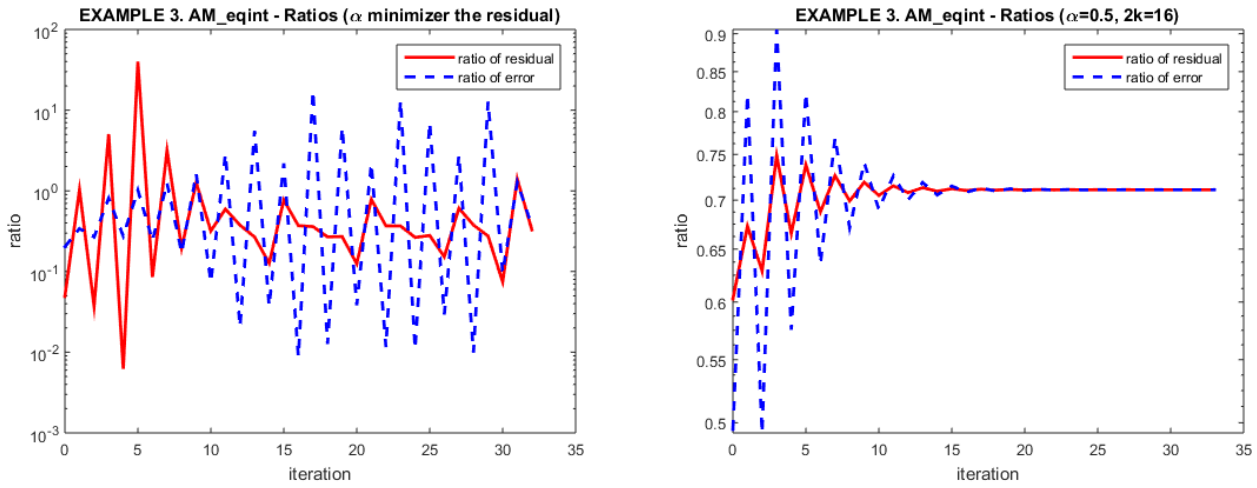

Figure 2.10: Example 3 - Comparison the ratios of AM_eqint between $\alpha = 0.1$ fixed strategy and α "minimizer the residual" strategy.

from the first iteration. In Figure 2.9 we plot the ratios for both cases. On the left we can recognize that "uncertainty" found in the first 8 iterations of the first case. On the right we can see that the better setting corresponds to the original sequence which has a constant behaviour of ratios.
For Example 3 we have shown an unemployable result for α "minimizer the residual" strategy with $2k = 10$. In Figure 2.10 (left) we plot the ratios and we note the same chaoticity found in Figure 2.8 (right). Finally for a fixed $\alpha = 0.55$ with $2k = 30$ that we have seen in Figure 2.7, we note that the extrapolation method is effective from the 6-th iteration; we can see in Figure 2.10 (right) that the ratios behaviour describes a triangle wave that rapidly dissolves and that is almost stable from the 6-th iteration.

We have just verified the Observation 2.1 for the Examples 2 and Example 3, and also we can noted a correlation between the two ratios in the Figure 2.9 and in the Figure 2.10. This suggests the following adaptive strategy for the step size $\alpha^{(n)}$ choice: let $\bar{r}$ be the **fixed ratio** we establish for every iteration the $\alpha^{(n+1)}$ such that

$$\frac{< \mathbf{y}, \mathbf{u}^{(n+1);\alpha^{(n+1)}} - \mathbf{T}[\mathbf{u}^{(n+1);\alpha^{(n+1)}}] >}{< \mathbf{y}, \mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}] >} = \bar{r} \tag{2.28}$$

In order to determinate it we use the secant method for the equivalent form

$$\underbrace{\alpha \; < \mathbf{y}, \mathbf{res}(n) > + < \mathbf{y}, \mathbf{T}\Big[\mathbf{u}^{(k)} - \alpha\, \mathbf{res}(n)\Big] > \; - < \mathbf{y}, \mathbf{u}^{(n)} > + \bar{r} \; < \mathbf{y}, \mathbf{res}(n) >}_{g(\alpha)} = 0$$

where $\mathbf{res}(n)$ indicates the precedent residual vector, i.e. $\mathbf{res}(n) := \mathbf{u}^{(n)} - \mathbf{T}[\mathbf{u}^{(n)}]$.

In numerical analysis, the secant method is a root-finding iterative algorithm given by

$$\alpha^{(i+1)} = \alpha^{(i)} - g(\alpha^{(i)})\frac{\alpha^{(i)} - \alpha^{(i-1)}}{g(\alpha^{(i)}) - g(\alpha^{(i-1)})}$$

where we denote with $(\alpha^{(i)})$ the sequence of scalar that converge to the $\alpha^{(n+1)}$ given by (2.28). We give in the Algorithm 2 the pseudo-code of secant method for our problem where we start[13] from $\alpha^{(0)} = 1$ and $\alpha^{(1)} = 0.95$ in order to favour a high step size. In the input there are the function g described in the previous page, the tolerance of the method Tol_Sec, and the maximum number of iterations Max_Iter_Sec for the stopping criterion[14].

Algorithm 2 Secant iterative method for determinate $\alpha^{(n+1)}$

Input: g, Tol_Sec, Max_Iter_Sec, $(\alpha^{(0)} = 1, \alpha^{(1)} = 0.95)$
$g\_value^{(0)} \leftarrow g(\alpha^{(0)})$
$g\_value^{(1)} \leftarrow g(\alpha^{(1)})$
$n \leftarrow 0$
while $g\_value^{(n)} >$ Tol_Sec **and** $n \leq$ Max_Iter_Sec **do**
    $n \leftarrow n + 1$
    $\alpha^{(n+1)} \leftarrow \alpha^{(n)} - g\_value^{(n)}\dfrac{\alpha^{(n)} - \alpha^{(n-1)}}{g\_value^{(n)} - g\_value^{(n-1)}}$
    $g\_value^{(n+1)} \leftarrow g(\alpha^{(n+1)})$
end while
Output: $\alpha^{(n+1)}$

From a practical point of view we "replace" the problem of choosing a fixed α with the problem of choosing a fixed ratio $\bar{r}$. But if the first one has an abstract sense for the user, the $\bar{r}$ represents the inclination of original sequence that will be similar to a descent straight. This aspect can be the upside of this strategy. However, it is more difficult to analytically study the interval in which the original sequence descent with the same inclination. Despite of the theoretical difficulty, from a practical point of view it is easy to automatize the selection process of $\bar{r}$ or to understand whether the wrong choice has been made as a consequence of the fact that we can study the behaviour of ratio of residual during the computation.

Now we show the numerical results[15] of our test examples. We determine the correct $\bar{r}$ favouring the terms closer to zero so that we make the original

[13]For determining the acceptable solution in fewer number of iterations, we can start from the precedent step size and a number close to it.

[14]In the experiments of the next pages we will set tolerance to 10^{-15}, and the maximum number of iterations to 100.

[15]For those experiments with our implementation we use the parameter: $\mathbf{y} \equiv 1$, Max_Iter = 50, and Tol= 10^{-15}.

sequence steeper. In this way, we obtain an original sequence with much more coherent informations; this means that we can increase $2k$ related to the number of terms used by the extrapolation method, and that the algorithm can converge more quickly. On the contrary, in the α fixed strategy when the parameter $2k$ is high the results are not as good as the one obtained with our adaptive strategy.

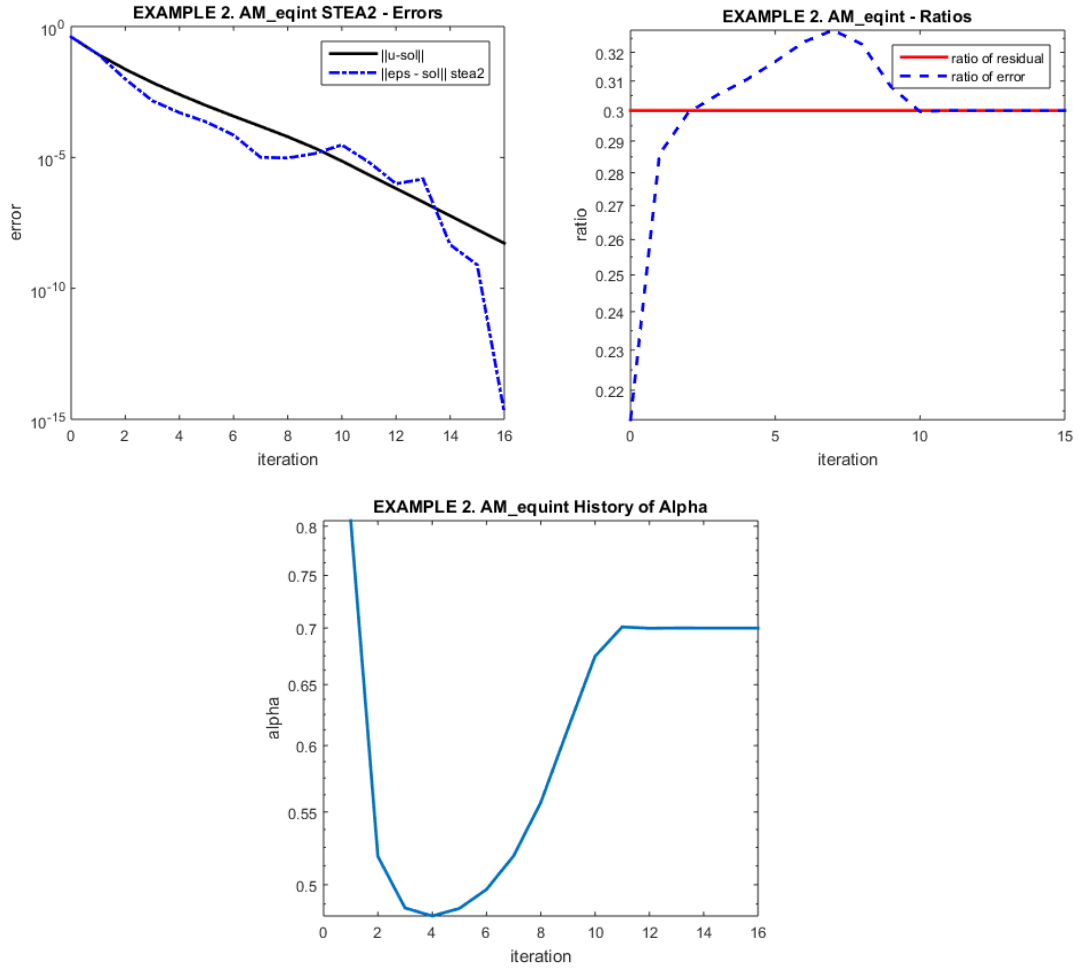

Figure 2.11: Example 2 - Representation of the errors, ratios and $\alpha^{(n)}$ obtained by AM_eqint according to adaptive strategy with ratio fixed to $\bar{r} = 0.3$ and with $2k = 10$.

In Figure 2.11 we show the performance of this adaptive strategy applied to Example 2 for $\bar{r} = 0.3$ and $2k = 10$. We note that until the 10-13-th iteration the extrapolation doesn't provide a really interesting performance because the ratio of error is not stationary. As we have already said, the ratio of error cannot be determined. However, what the Figure 2.11 suggests is that the convergence of ratio of the error is related to the steady state of $\alpha^{(n)}$ in these situations. Also for this property we have not succeeded to derive a theoretical proof; however the various experiments we have produced have always confirmed this property.

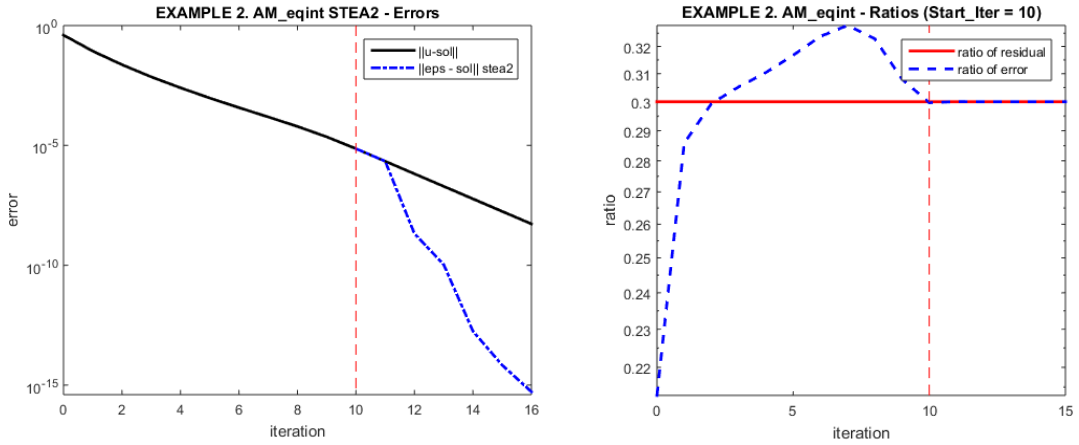

Figure 2.12: Example 2 - Representation of the errors, ratios and $\alpha^{(n)}$ obtained by AM_eqint according to adaptive strategy with ratio fixed to $\bar{r} = 0.3$ and with $2k = 10$ and starting iteration from 10-th iteration.

Therefore we try to use this property as a criterion to start the extrapolation, so that our algorithm becomes less expensive while maintaining its efficiency. In Figure 2.12 we show the behaviour of this strategy applied to the same Example 2 with the same last parameter but starting the `STEA2` algorithm from 10-th iteration. In the following Table 2.5 we report the complete results. We use a horizontal line between table records to highlight from which iteration numerical extrapolation was applied (when we start after first iteration).

	Original sequence					**Starting_Iter = 0**		**Starting_Iter = 10**	
n	**Residual**	**Error**	$\boldsymbol{\alpha^{(n)}}$	**Ratio of residual**	**Ratio of error**	**Residual**	**Error**	**Residual**	**Error**
0	3,90E-01	4,00E-01				3,90E-01	4,00E-01	3,90E-01	4,00E-01
1	1,20E-01	8,58E-02	8,06E-01	3,00E-01	2,13E-01	1,20E-01	8,58E-02	1,20E-01	8,58E-02
2	3,50E-02	2,33E-02	5,19E-01	3,00E-01	2,86E-01	1,32E-02	9,85E-03	3,50E-02	2,33E-02
3	9,84E-03	7,23E-03	4,85E-01	3,00E-01	2,99E-01	1,56E-03	1,46E-03	9,84E-03	7,23E-03
4	3,27E-03	2,53E-03	4,80E-01	3,00E-01	3,05E-01	4,78E-04	5,15E-04	3,27E-03	2,53E-03
5	1,15E-03	9,48E-04	4,85E-01	3,00E-01	3,10E-01	2,41E-04	2,25E-04	1,15E-03	9,48E-04
6	4,28E-04	3,76E-04	4,97E-01	3,00E-01	3,17E-01	7,96E-05	7,16E-05	4,28E-04	3,76E-04
7	1,65E-04	1,54E-04	5,19E-01	3,00E-01	3,24E-01	9,36E-06	1,01E-05	1,65E-04	1,54E-04
8	6,37E-05	6,18E-05	5,57E-01	3,00E-01	3,28E-01	8,73E-06	9,56E-06	6,37E-05	6,18E-05
9	2,29E-05	2,27E-05	6,13E-01	3,00E-01	3,23E-01	1,28E-05	1,41E-05	2,29E-05	2,27E-05
10	7,29E-06	7,29E-06	6,75E-01	3,00E-01	3,08E-01	2,84E-05	3,01E-05	7,29E-06	7,29E-06
11	2,18E-06	2,18E-06	7,01E-01	3,00E-01	3,00E-01	6,00E-06	6,62E-06	2,18E-06	2,18E-06
12	6,55E-07	6,55E-07	7,00E-01	3,00E-01	3,00E-01	9,04E-07	9,91E-07	2,05E-09	2,09E-09
13	1,96E-07	1,96E-07	7,00E-01	3,00E-01	3,00E-01	1,37E-06	1,50E-06	1,05E-10	1,07E-10
14	5,89E-08	5,89E-08	7,00E-01	3,00E-01	3,00E-01	6,87E-09	4,65E-09	1,84E-13	1,78E-13
15	1,77E-08	1,77E-08	7,00E-01	3,00E-01	3,00E-01	7,89E-10	7,95E-10	7,55E-15	6,83E-15
16	5,30E-09	5,30E-09	7,00E-01	3,00E-01	3,00E-01	1,72E-15	1,83E-15	5,55E-16	5,27E-16

Table 2.5: Example 2 - Performance of our adaptive step size strategy for different value of Starting_Iter with $\bar{r} = 0.3$, $\mathbf{y} \equiv 1$, Max_iter $= 50$, Tol$= 5 \cdot 10^{-15}$, using the Gaussian formula with $p = 8$, and starting from $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$.

Following the same arguments we arrive to similar results for Example 3. In the Figure 2.13 we show the result with $\bar{r} = 0.6$, $2k = 28$ and we start extrapolation from 10-th iteration. We have seen through our experiments that if we select a value of $\bar{r} < 0.6$ then we run into convergence problems. This phenomenon is closely linked to what is seen in Chapter 1 when, for the same example, we have seen that the convergence is not verified for $\alpha = 0.55$ $\forall n$. In Table 2.6 we report the complete results.

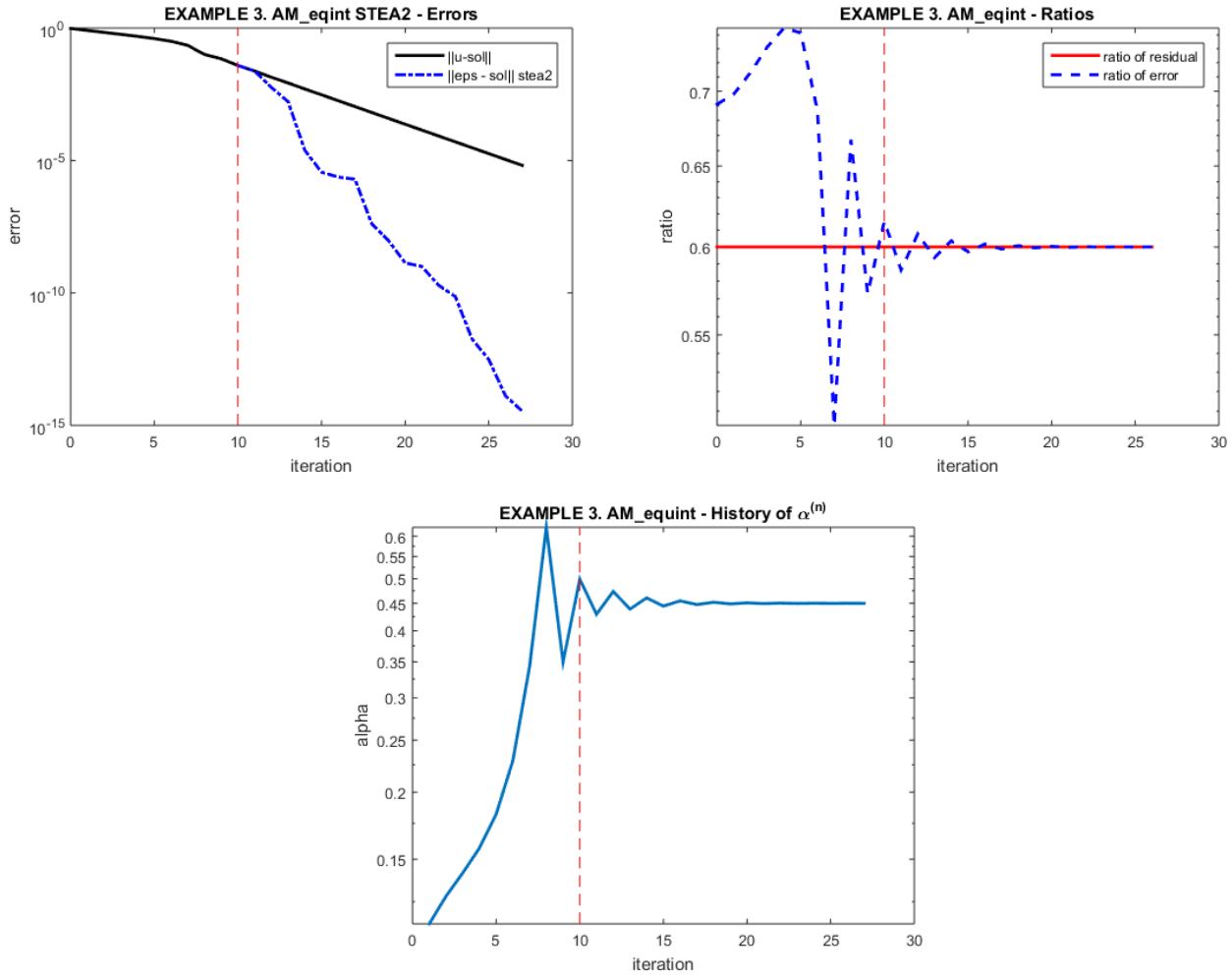

Figure 2.13: Example 3 - Representation of the errors, ratios and $\alpha^{(n)}$ obtained by AM_eqint according to adaptive strategy with ratio fixed to $\bar{r} = 0.6$ and with $2k = 28$.

	Original sequence					Starting_Iter = 0		Starting_Iter = 10	
n	**Residual**	**Error**	$\alpha^{(n)}$	**Ratio of residual**	**Ratio of error**	**Residual**	**Error**	**Residual**	**Error**
0	1,36E+00	9,66E-01				1,36E+00	9,66E-01	1,36E+00	9,66E-01
1	9,97E-01	8,12E-01	0,00E+00	6,00E-01	6,91E-01	9,97E-01	8,12E-01	9,97E-01	8,12E-01
2	7,55E-01	6,84E-01	1,14E-01	6,00E-01	6,98E-01	5,30E-01	4,19E-01	7,55E-01	6,84E-01
3	5,87E-01	5,77E-01	1,28E-01	6,00E-01	7,13E-01	4,01E-01	3,67E-01	5,87E-01	5,77E-01
4	4,63E-01	4,85E-01	1,41E-01	6,00E-01	7,31E-01	3,34E-01	3,14E-01	4,63E-01	4,85E-01
5	3,66E-01	4,01E-01	1,57E-01	6,00E-01	7,45E-01	3,41E-01	3,30E-01	3,66E-01	4,01E-01
6	2,81E-01	3,17E-01	1,82E-01	6,00E-01	7,42E-01	3,25E-01	3,87E-01	2,81E-01	3,17E-01
7	1,93E-01	2,21E-01	2,29E-01	6,00E-01	6,87E-01	2,74E-01	3,10E-01	1,93E-01	2,21E-01
8	9,02E-02	1,01E-01	3,43E-01	6,00E-01	5,03E-01	2,49E-01	3,11E-01	9,02E-02	1,01E-01
9	6,10E-02	6,91E-02	6,24E-01	6,00E-01	6,67E-01	2,18E-01	2,71E-01	6,10E-02	6,91E-02
10	3,44E-02	3,87E-02	3,50E-01	6,00E-01	5,73E-01	2,63E-01	3,17E-01	3,44E-02	3,87E-02
11	2,12E-02	2,39E-02	4,99E-01	6,00E-01	6,15E-01	1,64E-01	2,02E-01	2,12E-02	2,39E-02
12	1,23E-02	1,39E-02	4,29E-01	6,00E-01	5,87E-01	7,41E-02	9,09E-02	4,92E-03	5,74E-03
13	7,51E-03	8,46E-03	4,73E-01	6,00E-01	6,08E-01	1,99E-04	1,66E-04	1,46E-03	1,67E-03
14	4,45E-03	5,00E-03	4,39E-01	6,00E-01	5,94E-01	1,90E-05	1,87E-05	2,93E-05	2,43E-05
15	2,69E-03	3,02E-03	4,61E-01	6,00E-01	6,04E-01	2,53E-05	2,86E-05	8,78E-06	3,63E-06
16	1,60E-03	1,80E-03	4,45E-01	6,00E-01	5,97E-01	5,83E-06	4,17E-06	2,65E-06	2,36E-06
17	9,64E-04	1,08E-03	4,55E-01	6,00E-01	6,02E-01	2,13E-06	1,88E-06	2,24E-06	1,96E-06
18	5,77E-04	6,49E-04	4,47E-01	6,00E-01	5,99E-01	3,62E-08	4,26E-08	3,41E-08	4,07E-08
19	3,47E-04	3,90E-04	4,52E-01	6,00E-01	6,01E-01	1,18E-08	5,77E-09	1,39E-08	9,57E-09
20	2,08E-04	2,34E-04	4,49E-01	6,00E-01	5,99E-01	3,47E-09	1,36E-09	3,47E-09	1,36E-09
21	1,25E-04	1,40E-04	4,51E-01	6,00E-01	6,00E-01	2,16E-09	9,93E-10	2,16E-09	9,93E-10
22	7,48E-05	8,41E-05	4,49E-01	6,00E-01	6,00E-01	8,60E-11	9,82E-11	8,60E-11	9,82E-11
23	4,49E-05	5,05E-05	4,50E-01	6,00E-01	6,00E-01	1,89E-10	7,58E-11	1,89E-10	7,58E-11
24	2,69E-05	3,03E-05	4,50E-01	6,00E-01	6,00E-01	2,73E-12	2,49E-12	2,73E-12	2,49E-12
25	1,62E-05	1,82E-05	4,50E-01	6,00E-01	6,00E-01	2,47E-12	9,30E-13	2,47E-12	9,30E-13
26	9,69E-06	1,09E-05	4,50E-01	6,00E-01	6,00E-01	9,90E-14	8,64E-14	9,90E-14	8,64E-14
27	5,82E-06	6,54E-06	4,50E-01	6,00E-01	6,00E-01	7,14E-14	2,74E-14	7,14E-14	2,74E-14
28	3,49E-06	3,93E-06	4,50E-01	6,00E-01	6,00E-01	5,15E-15	7,23E-15	5,15E-15	7,23E-15
29	2,09E-06	2,36E-06	4,50E-01	6,00E-01	6,00E-01	7,92E-15	3,96E-15	7,92E-15	3,96E-15
30	1,26E-06	1,41E-06	4,50E-01	6,00E-01	6,00E-01	8,10E-15	3,66E-15		
31	7,54E-07	8,48E-07	4,50E-01	6,00E-01	6,00E-01	2,77E-15	5,60E-15		

Table 2.6: Example 3 - Performance of our adaptive step size strategy for different value of Starting_Iter with $\bar{r} = 0.6$, $\mathbf{y} \equiv 1$, $2k = 28$, Max_iter = 50, Tol = $5 \cdot 10^{-15}$, using the Gaussian formula with $p = 5$, and starting from $\mathbf{u}^{(0)} \equiv 1$.

We note that the performance of the fixed step size strategy is better than our adaptive strategy for the Example 2, as we can see by comparing the Table 2.3 and the Table 2.5. While for the Example 3, our adaptive strategy is better than results obtain with a fixed step size strategy studied in the previous section, as we can see by consulting the summary Table 2.4. In fact we don't see the phenomenon observed in Figure 2.7 (right) caused by cancellation errors and the AM_eqint converges on machine precision in 27 iterations where only 17 are extrapolated.
Therefore it seems that this adaptive strategy is of little use in the cases where the conditions of the Theorem 1.3 are satisfies. If the problem doesn't satisfy them, then this strategy could be effective from a practical point of view. In Chapter 3 we will see if this good property is also verified for other examples and we will evaluate more complex cases having a two-dimensional domain hoping to achieve a good performance compared to the state of the art.

Chapter 3

Comparing the fixed and the adaptive step size strategies

In the previous chapter we applied the simplified topological ε-algorithm [11] in order to accelerate the convergence of vector sequence $(\mathbf{u}^{(n)})$ generated by the Nyström method for solving the Urysohn integral equation,

$$u(t) = f(t) + \int_{\Omega} K(t, x, u(x))\, dx, \quad \text{for } t \in \Omega$$

We proved in Chapter 1 (see §1.2) that, under appropriate conditions, the error between the exact solution u^* and the n-th approximated solution

$$u^{(n)}(t) = f(t) + \sum_{i=0}^{p} w_i\, K(t, x_i, u_k^{(n)})$$

associated to the n-th vector of the sequence, $\mathbf{u}^{(n)} = (u_0^{(n)}, \ldots, u_p^{(n)})$, is bounded by the residual of $\mathbf{u}^{(n)}$ and the term $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ (related to the quadrature formula error according to the relations in Corollary 1.1). Thus, the determination of the solution minimizing the residual is as important as finding an appropriate quadrature formula in order to avoid the stagnation of the error.

We compute the original sequence $(\mathbf{u}^{(n)})$ by RPID where the element $\mathbf{u}^{(n+1)}$ is determined starting from $\mathbf{u}^{(n)}$ and moving along the current residual according to the step size $\alpha^{(n+1)}$. We compared two strategies for step size choice in the last section of Chapter 2. The first one is the fixed step size strategy where the step size is fixed to α, as the name suggests. In this case we choose the parameter $2k$ related to the number of elements involved to the extrapolation method. While in section 2.5 we present an adaptive strategy that could lead to an improvement of extrapolation performance. In order to improve the extrapolation performance with this strategy we have to avoid to start from the first iterations, we had rather start from a successive iteration according to the principle discussed in Chapter

2. We take the parameter $2k$ maximized[1], so that we can use the informations that we can obtain from the other iterations. Thus we must choose only the parameter $\bar{r}$. However we didn't take into account the choice of the parameter. From a practical point of view we saw two observation:

- in a problem where the conditions of Theorem 1.3 are satisfied, like Example 2, the choice $\alpha = 1$ and $2k$ maximized offers a good performance;
- in the cases in which the conditions of Theorem 1.3 are not satisfied, the adaptive strategy could be a better choice.

In the first section of this chapter we will see that the first observation is true for other examples, while we will show that for the best choice of ratio, there is an α which determines a better result with fixed step size strategy (for an appropriate choice of $2k$). We will also take the opportunity to observe that different choices of the dual vector $\mathbf{y}$ do not lead to different results for the first strategy while it sometimes plays a decisive role for the adaptive step size strategy. This will lead us to conclude that the idea proposed by Brezinski and Redivo-Zaglia in [10] is much more valid than an adaptive strategy. But in this thesis we made some steps ahead: a better theoretical understanding of the contribution due to the quadrature formula and a firm theorem through which we can fix $\alpha = 1$ without having problems of convergence.

Finally to demonstrate the effectiveness of what emerged in this thesis, we will see in the second section of this chapter how our method is able to solve Urysohn integral equations defined in multivariate domains and how our method is better than the solutions obtained by the authors of articles from which we took these examples.

For all numerical experiments contained in this Chapter we take $\text{Tol} = 5 \cdot 10^{-15}$, $\text{Tol\_Sec} = 10^{-15}$ and $\text{Max\_Iter\_Sec} = 100$.

3.1 Discussion about the parameters

In this section we discuss the choice of the parameters for both strategy. Initially we want to get convinced about the fact that the choice of $\alpha = 1$ and $2k$ maximized in those integral equations that satisfy the conditions of Theorem 1.3 is a good approach. Then we will analyse some examples in which these conditions are not satisfied and we will compare the two strategies, each of one with the best parameters.

[1]We conventionally decide that "2k maximized" stay for the involvement of all iteration in extrapolation method, i.e. $2k$ =Max_Iter−1 (if Max_Iter is even, otherwise we take $2k$ =Max_Iter−2).

Example 2

Let us consider the previous Example 2

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

and its the solution $u_2^*(t) = -\frac{1}{5}\cos\left(\frac{\pi t}{4}\right)$ that belongs to the confidence region $\widetilde{M} = [-\zeta, \zeta]$ with $\zeta \in \left(\frac{1}{4}, \frac{4}{3\sqrt{2}\pi}\right)$. For our tests we take $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2}\pi}$, Max_Iter $= 20$ and we use the Gaussian quadrature formula with $p = 8$ in order to reach the machine precision, as we have seen before.

We remind that the vector $\mathbf{w} = (w_0, \ldots, w_p)$ denotes the weight of the quadrature formula that we have chosen. In Section 2.4 we observed that in the Example 2 the performance of fixed step size strategy with $\alpha = 1$ and $2k$ maximized gives better results respect to these other choices: $\alpha = 0.5$ (for $2k$ maximized and $2k = 4$) or only $\alpha = 1$ (for $2k = 4$). In reality this setting is the best among all the possible parameters of $2k = 2, 4, \ldots$ and $\alpha \in (0, 1]$. In order to get convinced of this, in the Appendix A, Table A.1 we show the number of iterations within which the algorithm stops and the accuracy reached for each combination of $\alpha = 0.1, 0.2, \ldots, 1$ and $2k$. We have used the dual vector $\mathbf{y} \equiv 1$ in order to compute the values of this table, but there are only small differences when we choose $\mathbf{y} = \mathbf{w}$, as we can see in Table A.2 where we employed $\mathbf{y} = \mathbf{w}$. Brezinski and Redivo-Zaglia [10] obtained in their examples a better performance using a dual vector with components randomly generated in $(-1, 1)$. In our experiments we have never noticed this, as we can see in the following table where we have compared the three different dual vectors: $\mathbf{y} \equiv 1$, $\mathbf{y} = \mathbf{w}$ and $\mathbf{y}$ random.

	y ≡ 1		**y = *w***		**y random**	
	Residual	**Error**	**Residual**	**Error**	**Residual**	**Error**
0	3,90E-01	4,00E-01	3,90E-01	4,00E-01	4,00E-01	4,00E-01
1	1,49E-02	1,00E-02	1,49E-02	1,00E-02	1,00E-02	1,00E-02
2	8,20E-03	5,44E-03	8,22E-03	5,46E-03	5,47E-03	5,46E-03
3	6,17E-05	4,11E-05	6,17E-05	4,11E-05	4,11E-05	4,11E-05
4	1,51E-05	1,01E-05	1,51E-05	1,01E-05	1,01E-05	1,01E-05
5	2,48E-07	1,65E-07	2,48E-07	1,65E-07	1,65E-07	1,65E-07
6	3,14E-08	2,10E-08	3,14E-08	2,10E-08	2,10E-08	2,10E-08
7	1,95E-10	1,30E-10	1,95E-10	1,30E-10	1,30E-10	1,30E-10
8	1,20E-11	8,01E-12	1,20E-11	8,01E-12	8,01E-12	8,01E-12
9	5,27E-14	3,52E-14	5,27E-14	3,52E-14	3,52E-14	3,52E-14
10	1,78E-15	1,17E-15	1,67E-15	1,08E-15	1,14E-15	1,08E-15

Table 3.1: Example 2 - Performance of the fixed step size strategy with better choice of parameters: $\alpha = 1$ and $2k$ maximized for different dual vectors $\mathbf{y}$.

In order to compare the fixed step size strategy with adaptive step size strategy, we apply the same methodology to determine the best parameter ratio for the latter strategy. To make the result more accessible we put in columns the results

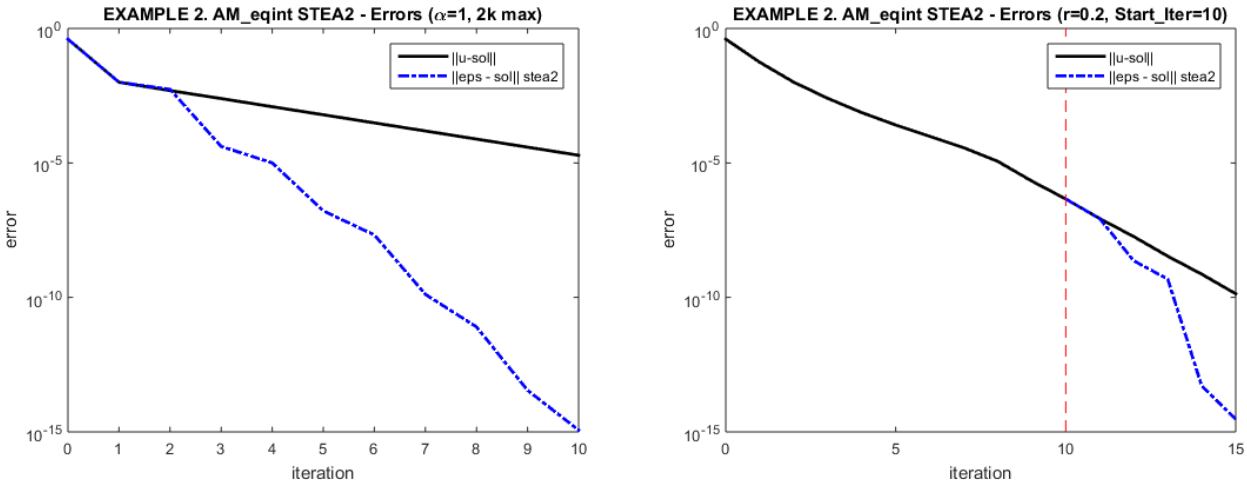

Figure 3.1: Example 2 - Comparison between the two strategies, each of one with the best parameters. Left: fixed step size strategy with $\alpha = 1$ and $2k$ maximized, using $\mathbf{y} \equiv 1$. Right: adaptive step size strategy with $\bar{r} = 0.2$ and Starting_Iteration $= 10$, using $\mathbf{y} = \mathbf{w}$.

for different Starting_Iterations number, i.e. starting the extrapolation method from different iterations. Therefore we can clearly see the best ratio (and it best starting_iteration) using the Table B.2, in Appendix B. We obtain the best result with $\bar{r} = 0.2$. In Figure 3.1 we compare the error between the two strategies, each of one with their best parameters and in the following Table 3.2 we report the numerical results of adaptive step size strategy. We see that the adaptive original sequence have a better convergence respect to the sequence gererated by Picard iteration (used in the fixed strategy). We note also the two extrapolated sequence have a quite similar behaviour.

	Original sequence					**Starting_Iter = 0**		**Starting_Iter = 10**	
n	**Residual**	**Error**	$\boldsymbol{\alpha^{(n)}}$	**Ratio of residual**	**Ratio of error**	**Residual**	**Error**	**Residual**	**Error**
0	3,90E-01	4,00E-01	0,00E+00			3,90E-01	4,00E-01	3,90E-01	4,00E-01
1	7,99E-02	5,53E-02	8,84E-01	2,00E-01	1,37E-01	7,99E-02	5,53E-02	7,99E-02	5,53E-02
2	1,54E-02	1,00E-02	5,67E-01	2,00E-01	1,93E-01	6,18E-03	4,70E-03	1,54E-02	1,00E-02
3	3,29E-03	2,52E-03	5,41E-01	2,00E-01	2,00E-01	7,60E-04	7,39E-04	3,29E-03	2,52E-03
4	8,93E-04	7,44E-04	5,40E-01	2,00E-01	2,03E-01	2,87E-04	2,94E-04	8,93E-04	7,44E-04
5	2,84E-04	2,57E-04	5,45E-01	2,00E-01	2,08E-01	1,18E-04	1,22E-04	2,84E-04	2,57E-04
6	1,02E-04	9,76E-05	5,61E-01	2,00E-01	2,16E-01	2,26E-05	2,33E-05	1,02E-04	9,76E-05
7	3,72E-05	3,67E-05	5,98E-01	2,00E-01	2,28E-01	9,66E-06	9,90E-06	3,72E-05	3,67E-05
8	1,14E-05	1,14E-05	6,80E-01	2,00E-01	2,29E-01	7,40E-06	7,58E-06	1,14E-05	1,14E-05
9	2,13E-06	2,13E-06	8,15E-01	2,00E-01	1,95E-01	1,08E-05	1,10E-05	2,13E-06	2,13E-06
10	4,57E-07	4,57E-07	7,84E-01	2,00E-01	2,05E-01	3,97E-06	4,07E-06	4,57E-07	4,57E-07
11	8,52E-08	8,51E-08	8,14E-01	2,00E-01	1,95E-01	4,39E-06	4,50E-06	8,52E-08	8,51E-08
12	1,82E-08	1,83E-08	7,85E-01	2,00E-01	2,05E-01	5,42E-07	5,55E-07	2,26E-09	2,30E-09
13	3,41E-09	3,41E-09	8,14E-01	2,00E-01	1,95E-01	1,62E-08	1,66E-08	4,74E-10	4,81E-10
14	7,30E-10	7,30E-10	7,85E-01	2,00E-01	2,05E-01	7,85E-10	7,94E-10	4,96E-14	5,13E-14
15	1,37E-10	1,36E-10	8,14E-01	2,00E-01	1,95E-01	7,30E-11	7,34E-11	2,75E-15	2,97E-15
16	2,92E-11	2,92E-11	7,85E-01	2,00E-01	2,05E-01	2,87E-12	2,89E-12		

Table 3.2: Example 2 - Performance of the adaptive step size strategy for the best choice of parameters: $\bar{r} = 0.2$ and Start_Iterations $= 10$ using $\mathbf{y} = \mathbf{w}$.

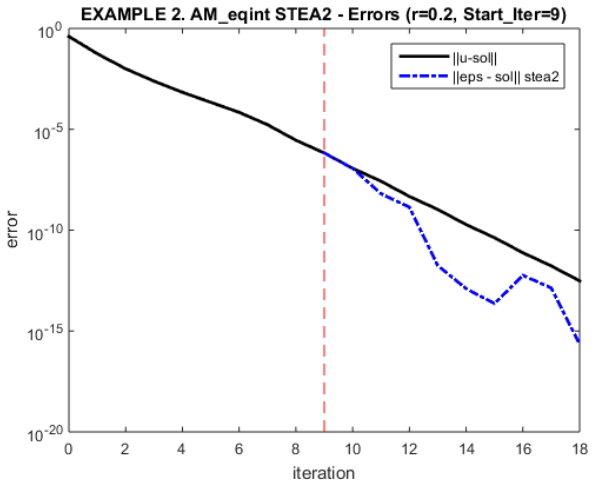

Figure 3.2: Example 2 - The error behaviour for the adaptive step size strategy with $\bar{r} = 0.2$, Start_Iterations $= 9$, and $\mathbf{y} \equiv 1$.

For computing the value of Table 3.2 we have used the dual vector $\mathbf{y} = \mathbf{w}$. In fact during our investigation we have establish that $\mathbf{y} = \mathbf{w}$ produces sometimes more stable results. In Table 3.3 we show the numerical results obtained with $\mathbf{y} \equiv 1$. In Figure 3.2 we observe that the delayed extrapolation is not effective using the $\mathbf{y} \equiv 1$. In Table 3.3 we see that the strategy with Starting_Iteration $= 0$ finishes in only 16 iterations. However we want to exploit the delayed extrapolation of adaptive strategy in order to reduce the number of extrapolated iterations.

	Original sequence					**Starting_Iter = 0**		**Starting_Iter = 9**	
n	**Residual**	**Error**	$\alpha^{(n)}$	**Ratio of residual**	**Ratio of error**	**Residual**	**Error**	**Residual**	**Error**
0	3,90E-01	4,00E-01				3,90E-01	4,00E-01	3,90E-01	4,00E-01
1	8,07E-02	5,59E-02	8,82E-01	2,00E-01	1,38E-01	8,07E-02	5,59E-02	8,07E-02	5,59E-02
2	1,53E-02	1,00E-02	5,69E-01	2,00E-01	1,94E-01	5,93E-03	4,53E-03	1,53E-02	1,00E-02
3	3,17E-03	2,44E-03	5,47E-01	2,00E-01	2,04E-01	6,14E-04	6,38E-04	3,17E-03	2,44E-03
4	8,14E-04	6,86E-04	5,52E-01	2,00E-01	2,11E-01	2,36E-04	2,34E-04	8,14E-04	6,86E-04
5	2,35E-04	2,18E-04	5,75E-01	2,00E-01	2,22E-01	7,83E-05	7,36E-05	2,35E-04	2,18E-04
6	7,03E-05	6,94E-05	6,31E-01	2,00E-01	2,32E-01	1,28E-05	1,39E-05	7,03E-05	6,94E-05
7	1,68E-05	1,69E-05	7,46E-01	2,00E-01	2,16E-01	1,19E-05	1,31E-05	1,68E-05	1,69E-05
8	2,90E-06	2,87E-06	8,36E-01	2,00E-01	1,87E-01	1,62E-05	1,77E-05	2,90E-06	2,87E-06
9	6,66E-07	6,70E-07	7,59E-01	2,00E-01	2,13E-01	2,93E-07	3,00E-07	6,66E-07	6,70E-07
10	1,17E-07	1,16E-07	8,31E-01	2,00E-01	1,89E-01	2,62E-07	2,86E-07	1,17E-07	1,16E-07
11	2,65E-08	2,67E-08	7,65E-01	2,00E-01	2,11E-01	8,43E-07	9,25E-07	6,04E-09	6,44E-09
12	4,73E-09	4,70E-09	8,28E-01	2,00E-01	1,90E-01	1,02E-08	1,09E-08	1,29E-09	1,38E-09
13	1,06E-09	1,06E-09	7,68E-01	2,00E-01	2,10E-01	1,05E-09	1,11E-09	1,93E-12	1,76E-12
14	1,90E-10	1,89E-10	8,26E-01	2,00E-01	1,91E-01	3,60E-11	3,82E-11	1,32E-13	1,26E-13
15	4,22E-11	4,24E-11	7,71E-01	2,00E-01	2,09E-01	5,55E-14	5,32E-14	3,12E-14	2,30E-14
16	7,65E-12	7,61E-12	8,24E-01	2,00E-01	1,92E-01	3,89E-15	3,72E-15	5,98E-13	5,77E-13
17								1,41E-13	1,37E-13
18								1,94E-16	2,22E-16

Table 3.3: Example 2 - Performance of the adaptive step size strategy for the best choice of parameters: $\bar{r} = 0.2$ and Start_Iterations $= 9$ using $\mathbf{y} \equiv 1$.

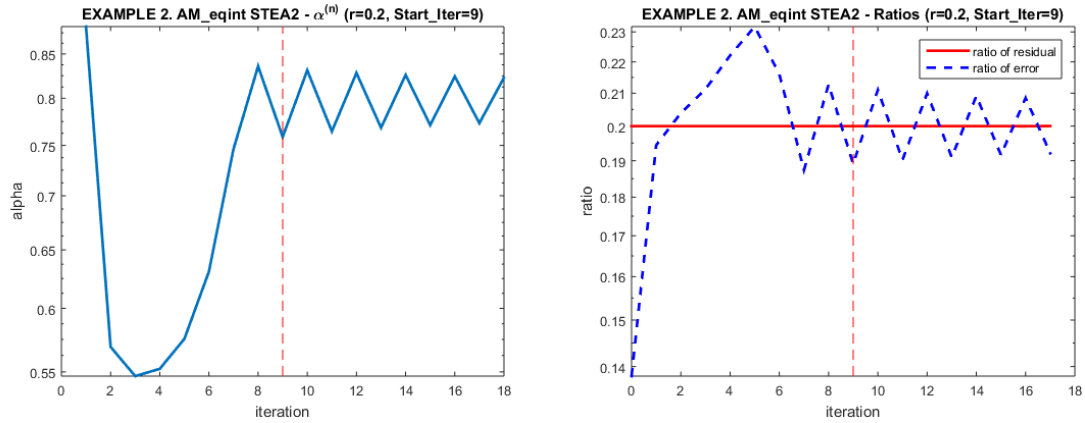

Figure 3.3: Example 2 - Performance of the adaptive step size strategy for the best choice of parameters: $\bar{r} = 0.2$ and Start_Iterations $= 9$ using the dual vector $\mathbf{y} \equiv 1$.

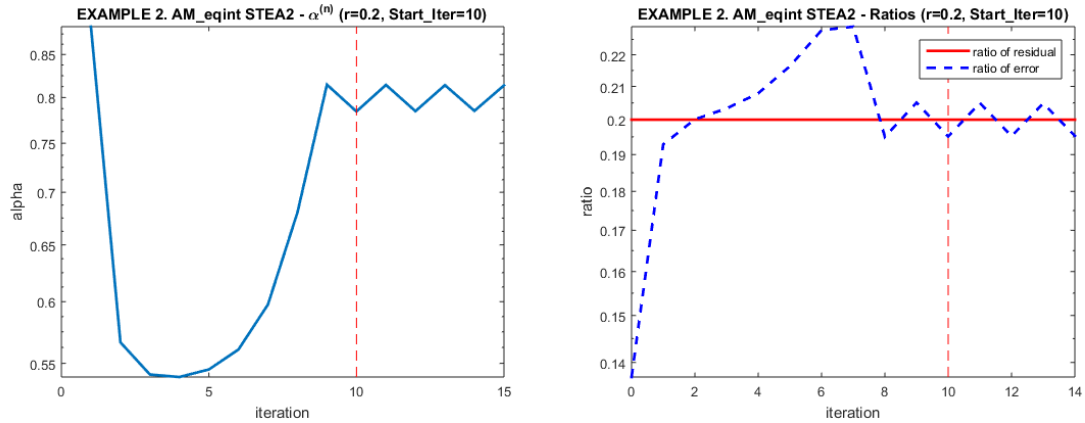

Figure 3.4: Example 2 - Performance of the adaptive step size strategy for the best choice of parameters: $\bar{r} = 0.2$ and Start_Iterations $= 10$ using $\mathbf{y} = \mathbf{w}$.

In order to give a more complete representation of what happens for $\mathbf{y} = \mathbf{w}$ and $\mathbf{y} \equiv 1$ in the adaptive step size strategy we plot the $\alpha^{(n)}$ and the ratios behaviour in Figure 3.3 and Figure 3.4. We note that the two setting show a quite similar behaviour but the oscillation for $\mathbf{y} \equiv 1$ is higher than the former. It is possible that this makes $\mathbf{y} \equiv 1$ less effective than $\mathbf{y} = \mathbf{w}$.

In the next example we can review the same aspects, therefore we will only provide a simple display of the numerical results.

Example 4

For a direct comparison between our performance and that obtained in [10], we study the Example 4 where the conditions of the Theorem 1.3 are satisfied. In [10, Example 4], Brezinski and Redivo-Zaglia presented the following equation

$$u(t) = \frac{t}{20}\int_0^1 x\,u^2(x)\,dx + 3 + 0.6625\,t$$

in which its solution is $u^*(x) = x + 3$ that belongs to the confidence region $\widetilde{M} = [-\zeta, \zeta]$ with $\zeta = [4.07832, 10)$. For our experiments we take $\mathbf{u}^{(0)} \equiv 1$, Max_Iter = 30 and we use the Gaussian quadrature formula with $p = 6$ because we have $\mathcal{E}^{\mathrm{QUAD}}(u^*) \sim 6.939 \cdot 10^{-16}$.

In Table A.5 (Appendix A) we can see that $\alpha = 1$ and $2k$ maximized is the best choice for fixed step size strategy, while in the Table B.6 (Appendix B) we note that $\bar{r} = 0.1$ and init_iter = 2 is the best choice for adaptive step size strategy. In the Tables 3.4, 3.5, 3.6 we report the numerical results and in the Figure 3.5 we compare the behaviour of the error between these strategies.

	y ≡ 1		**y = *w***		**y random**	
n	**Residual**	**Error**	**Residual**	**Error**	**Residual**	**Error**
0	2,67E+00	2,97E+00	2,67E+00	2,97E+00	2,67E+00	2,97E+00
1	2,68E-01	3,05E-01	2,68E-01	3,05E-01	2,68E-01	3,05E-01
2	1,77E-02	2,02E-02	1,77E-02	2,02E-02	8,24E-02	9,40E-02
3	1,43E-04	1,64E-04	1,43E-04	1,64E-04	1,43E-04	1,64E-04
4	1,65E-06	1,89E-06	1,65E-06	1,89E-06	2,55E-06	2,91E-06
5	2,11E-09	2,42E-09	2,11E-09	2,42E-09	2,11E-09	2,42E-09
6	3,41E-12	3,89E-12	3,41E-12	3,89E-12	4,36E-12	4,98E-12
7	1,33E-15	1,78E-15	8,88E-16	1,33E-15	8,88E-16	8,88E-16

Table 3.4: Example 4 - Performance of the fixed step size strategy with better choice of parameters: $\alpha = 1$ and $2k$ maximized for different dual vectors $\mathbf{y}$.

	Original sequence					**Starting_Iter = 0**		**Starting_Iter = 10**	
n	**Residual**	**Error**	$\boldsymbol{\alpha^{(n)}}$	**Ratio of residual**	**Ratio of error**	**Residual**	**Error**	**Residual**	**Error**
0	2,67E+00	2,97E+00	0,00E+00			2,67E+00	2,97E+00	2,67E+00	2,97E+00
1	3,70E-01	4,26E-01	9,55E-01	1,00E-01	1,05E-01	3,70E-01	4,26E-01	3,70E-01	4,26E-01
2	4,63E-02	5,28E-02	1,01E+00	1,00E-01	1,02E-01	1,09E-02	1,09E-02	0,046297	5,28E-02
3	4,55E-03	5,20E-03	1,03E+00	1,00E-01	1,00E-01	1,95E-04	2,18E-04	0,004551	5,20E-03
4	4,57E-04	5,23E-04	1,03E+00	1,00E-01	1,00E-01	1,76E-05	1,95E-05	2,85E-06	3,09E-06
5	4,57E-05	5,22E-05	1,03E+00	1,00E-01	1,00E-01	8,03E-08	8,80E-08	9,45E-08	1,03E-07
6	4,57E-06	5,22E-06	1,03E+00	1,00E-01	1,00E-01	1,17E-09	1,28E-09	2,41E-09	2,64E-09
7	4,57E-07	5,22E-07	1,03E+00	1,00E-01	1,00E-01	5,98E-12	6,56E-12	1,34E-11	1,46E-11
8	4,57E-08	5,22E-08	1,03E+00	1,00E-01	1,00E-01	4,80E-14	5,20E-14	4,00E-15	3,55E-15
9	4,57E-09	5,22E-09	1,03E+00	1,00E-01	1,00E-01	1,78E-15	2,22E-15		

Table 3.5: Example 4 - Performance of the adaptive step size strategy for the best choice of parameters: $\bar{r} = 0.1$ and Start_Iterations = 2 using $\mathbf{y} = \mathbf{w}$.

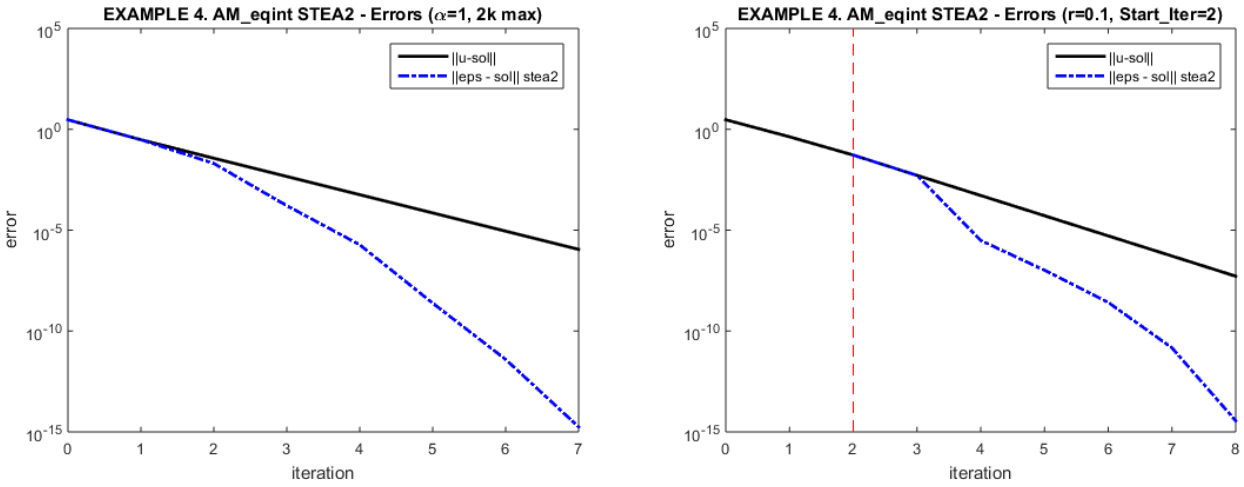

Figure 3.5: Example 4 - Comparison between the two strategies, each of one with the best parameters. Left: fixed step size strategy with $\alpha = 1$ and $2k$ maximized, using $\mathbf{y} \equiv 1$. Right: adaptive step size strategy with $\bar{r} = 0.1$ and Starting_Iteration $= 2$, using $\mathbf{y} = \mathbf{w}$.

	Original sequence					**Starting_Iter = 0**		**Starting_Iter = 2**	
n	**Residual**	**Error**	$\boldsymbol{\alpha^{(n)}}$	**Ratio of residual**	**Ratio of error**	**Residual**	**Error**	**Residual**	**Error**
0	2,67E+00	2,97E+00				2,67E+00	2,97E+00	2,67E+00	2,97E+00
1	3,70E-01	4,26E-01	9,55E-01	1,00E-01	1,05E-01	3,70E-01	4,26E-01	3,70E-01	4,26E-01
2	4,63E-02	5,28E-02	1,01E+00	1,00E-01	1,02E-01	1,09E-02	1,09E-02	4,63E-02	5,28E-02
3	4,55E-03	5,20E-03	1,03E+00	1,00E-01	1,00E-01	1,95E-04	2,18E-04	4,55E-03	5,20E-03
4	4,57E-04	5,23E-04	1,03E+00	1,00E-01	1,00E-01	1,76E-05	1,95E-05	2,85E-06	3,09E-06
5	4,57E-05	5,22E-05	1,03E+00	1,00E-01	1,00E-01	8,03E-08	8,80E-08	9,45E-08	1,03E-07
6	4,57E-06	5,22E-06	1,03E+00	1,00E-01	1,00E-01	1,17E-09	1,28E-09	2,41E-09	2,64E-09
7	4,57E-07	5,22E-07	1,03E+00	1,00E-01	1,00E-01	5,98E-12	6,56E-12	1,34E-11	1,46E-11
8	4,57E-08	5,22E-08	1,03E+00	1,00E-01	1,00E-01	4,71E-14	5,11E-14	3,11E-15	3,55E-15
9	4,57E-09	5,22E-09	1,03E+00	1,00E-01	1,00E-01	8,88E-16	1,33E-15		

Table 3.6: Example 4 - Performance of the adaptive step size strategy for the best choice of parameters: $\bar{r} = 0.1$ and Start_Iterations $= 2$ using the dual vector $\mathbf{y} \equiv 1$.

In this case we observe that there are many values not reported in the Table B.1 (Appendix B), where we use $\mathbf{y} \equiv 1$ instead $\mathbf{y} = \mathbf{w}$. This means that for some $\bar{r}$ the method not converge and in general that often happens when we take $\mathbf{y} \equiv 1$. Therefore unlike the fixed strategy, the adaptive step size strategy is influenced by the choice of $\mathbf{y}$.

Also in this case we see that the performances are similar between the two methods. However we cannot forget the computational cost necessary to the adaptive strategy in order to get the appropriate value of the step size.

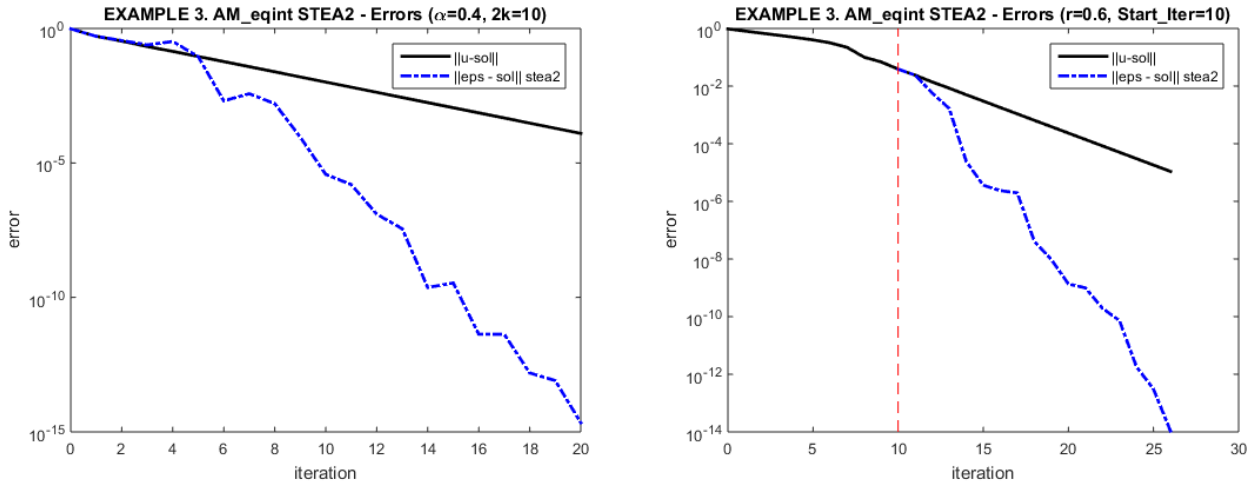

Figure 3.6: Example 3 - Comparison between the two strategies, each of one with the best parameters. Left: fixed step size strategy with $\alpha = 0.4$ and $2k = 10$, using $\mathbf{y} \equiv 1$. Right: adaptive step size strategy with $\bar{r} = 0.6$ and Starting_Iteration $= 10$, using $\mathbf{y} = \mathbf{w}$.

Example 3

Let us reuse the previous Example 3

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

where we consider the solution $u^*(t) = t$. We know that the conditions of Theorem 1.3 are not satisfied because the kernel of integral equation is exponential respect to the variable u. For our tests we take $\mathbf{u}^{(0)} \equiv 1$, Max_Iter $= 50$ and we use the Gaussian quadrature formula with $p = 5$ in order to reach the machine precision, as we have seen before.
In the section §2.5 we showed that the adaptive step size strategy with ratio $\bar{r} = 0.6$ is better than the fixed step size strategy obtained by $2k = 30$ and using the maximum step size which guaranties the convergence of the original sequence, $\alpha = 0.55$. In the Table B.4 (Appendix B) we can see that $\bar{r} = 0.6$ is the best choice for adaptive step size strategy. While in Table A.3 (Appendix A) we observe that $\alpha = 0.4$ and $2k = 10$ is the best choice for fixed step size strategy and with these parameters we obtain a better result respect to the best adaptive strategy. In the Figure 3.6 we compare graphically the behaviour of the error for both strategy and in the Table 3.7 we propose the performance of the best fixed step size strategy while in the previous Table 2.6 (page 55) we have seen the performance of the best adaptive strategy[2].

[2]For this example the performance of the best adaptive step size strategy using $\mathbf{y} \equiv 1$ is the same of that obtained using $\mathbf{y} = \mathbf{w}$ (reported in Table 2.6).

In this case the fixed strategy not converges using a random vector and we note that this strategy converges in a better way with $\mathbf{y} \equiv 1$ rather than with to $\mathbf{y} = \mathbf{w}$. In reality the phenomenon that is observed in the Table 3.7 is due to cancellation errors, it is usual when we approach the precision of the machine. Therefore we can rule out a preferential choice for the dual vector $\mathbf{y}$ suggested by this observation.

	$\mathbf{y} \equiv 1$		$\mathbf{y} = w$	
n	**Residual**	**Error**	**Residual**	**Error**
0	1,36E+00	9,66E-01	1,36E+00	9,66E-01
1	8,56E-01	5,10E-01	8,56E-01	5,10E-01
2	2,92E-01	3,56E-01	2,92E-01	3,56E-01
3	2,15E-01	2,44E-01	2,15E-01	2,44E-01
4	2,87E-01	3,28E-01	2,87E-01	3,28E-01
5	7,42E-02	8,90E-02	7,42E-02	8,90E-02
6	2,51E-03	2,03E-03	2,51E-03	2,03E-03
7	2,98E-03	3,74E-03	2,98E-03	3,74E-03
8	0,001299	0,001604	0,001299	0,001604
9	7,28E-05	9,09E-05	7,28E-05	9,09E-05
10	3,23E-06	3,75E-06	3,23E-06	3,75E-06
11	1,41E-06	1,57E-06	1,41E-06	1,57E-06
12	1,26E-07	1,24E-07	1,26E-07	1,24E-07
13	3,30E-08	3,57E-08	3,30E-08	3,57E-08
14	2,15E-10	2,34E-10	2,15E-10	2,34E-10
15	3,19E-10	3,45E-10	3,19E-10	3,45E-10
16	3,74E-12	4,29E-12	3,72E-12	4,27E-12
17	4,02E-12	4,26E-12	4,01E-12	4,24E-12
18	1,78E-13	1,55E-13	1,81E-13	1,56E-13
19	7,93E-14	8,09E-14	8,07E-14	8,38E-14
20	6,66E-16	2,21E-15	1,30E-14	1,34E-14
21			3,06E-14	4,01E-14
22			3,44E-14	4,03E-14
23			1,03E-13	1,32E-13
24			5,32E-15	9,04E-15
25			1,11E-15	1,78E-15

Table 3.7: Example 3 - Performance of the fixed step size strategy with better choice of parameters: $\alpha = 0.4$ and $2k = 10$ for different dual vectors $\mathbf{y}$.

In the Chapter 2 we hypothesized that the adaptive step size strategy could have a better behaviour than the fixed step size strategy in the study of integral equations in which the conditions of Theorem 1.3 are not satisfied. But taking the same example that which led us to suppose that, and using the better parameters for fixed step size strategy, we obtain a better result than the one we have achieved in Table 2.6 by adaptive step size strategy with the best parameters.

For this reason we consider plausible that the adaptive strategy does not offer an effective advantage but it is easier to employ for users respect to the fixed step size strategy in the present implementation. In the conclusion §3.3 we will discuss an idea to make also the fixed step size strategy more accessible for users.

3.2 Examples in bivariate domain

In this section we want to analyse the behaviour of our algorithm for the Urysohn integral equation for domain bivariate, a novel application never considered before. When it's possible we compare our algorithm to the precision obtained by the authors of other articles.

We have observed in the univariate case that the choice of an adequate quadrature formula has important consequences in terms of the error. For the univariate case the literature is exhaustive in determining quadrature formulas for a interval $[a, b]$ but for a general multivariate case a preferred choice does not exist and even elementary forms of algebraic quadrature formulas are unknown. For this reason we will study only the Urysohn integral equations that are defined on domains like square and disk for which we have available software sets or packages to determine an appropriate quadrature formula. We note that this limit imposed by the use of a Nyström method can be overcome only by changing the resolving approach. For example Assari and Dehghan explore in their recent article [4] the *thin plate spline collocation* method and they perform the result for domains of forms of which it would be difficult to determine a quadrature formula.

3.2 Examples in square

The most recurrent multivariate integral equation in engineering articles are defined on square. Many efforts have recently been made and many of them are based on the Nyström method. As we have already analysed in Chapter 1, the error is directly influenced by the quadrature formula. We remind the quadrature formula for bivariate domain is a weighted sum of function values at specified points $\{(x_0, y_0), \ldots, (x_p, y_p)\}$. We synthetically indicates that by the tern $(x_i, y_i, w_i)_{i=0,\ldots,p}$.

In our experiments we use the set of minimal quadrature formulas for low degree identified by Sommariva [28] for the square $[-1, 1] \times [-1, 1]$. When our square has a general form $[a, b] \times [a, b]$, we transform the Sommariva's quadrature formula through the following linear transformation. Let $(x_i, y_i, w_i)_{i=0,\ldots,p}$ the quadrature formula on $[-1, 1] \times [-1, 1]$, we define the quadrature formula $(\tilde{x}_i, \tilde{y}_i, \tilde{w}_i)_{i=0,\ldots,p}$ on $[a, b] \times [a, b]$ given by

$$\widetilde{x}_i := a + (x_i + 1)\frac{b-a}{2} \qquad \tilde{y}_i := a + (y_i + 1)\frac{b-a}{2} \qquad \tilde{w}_i := w_i \frac{(b-a)^2}{4}$$

For every example presented in this section, we identify a useful quadrature formula from the set [28] testing the term $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ for different degree. We show that our algorithm reaches the machine precision for every example in this section in contrast to the numerical methods studied in the articles from which they are

drawn. We do not have the source code of these methods in order to compare the computation time, but we consider plausible that the time required is comparable or bigger than our proposals.

Example 20

We consider the example presented in [16] where the authors obtain a precision of magnitude order 10^{-10} with Nyström Method accelerated by Richardson extrapolation. The example studies the integral equation

$$u(t,s) = -\log\left(1+\frac{s\,t}{1+s^2}\right) + \frac{t}{16\,(1+y)} + \int_{[0,1]\times[0,1]} \frac{t\,(1-x^2)}{(1+s)\,(1+y^2)}\left(1-e^{-u(x,y)}\right)\,dxy$$

where a solution is $u^*(t,s) = -log\left(1+\frac{s\,t}{1+s^2}\right)$. The quadrature formula of degree 24 given by [28] determines a $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ close to machine precision and we observe that the conditions of the Theorem 1.3 are not satisfied by this integral equation. In fact we have

$$\max_{0\le t,s,x,y\le 1}\left|\frac{\partial}{\partial u}\left[\frac{t\,(1-x^2)}{(1+s)\,(1+y^2)}\left(1-e^{-u}\right)\right]\right| = e^{-u}$$

and this is less than 1 for $u > 0$. Also the iv) condition that we introduced in Theorem 1.3, to guarantee to stay in the confidence zone, is not satisfied. Nevertheless, we will test our algorithms also in this case, hoping that they will converge to the solution indicated by [16] starting from an arbitrary vector, like $\mathbf{u}^{(0)} \equiv 1$.

	Fixed step size strategy $\boldsymbol{\alpha = 1}$ **Num_Iter = 11 2k=10**				**Adaptive step size strategy** $\boldsymbol{\bar{r} = 0,1}$ **Num_Iter = 13 2k=12 y = w**				
	y ≡ 1		**y = w**			**Start_Iter = 0**		**Start_Iter = 4**	
n	**Residual**	**Error**	**Residual**	**Error**	$\boldsymbol{\alpha^{(n)}}$	**Residual**	**Error**	**Residual**	**Error**
0	1,20E+00	1,40E+00	1,20E+00	1,40E+00		1,20E+00	1,40E+00	1,20E+00	1,40E+00
1	3,25E-01	3,83E-01	3,25E-01	3,83E-01	1,01687	3,23E-01	3,72E-01	3,23E-01	3,72E-01
2	1,23E-02	1,47E-02	1,35E-02	1,61E-02	1,05206	7,19E-03	6,95E-03	2,66E-02	3,22E-02
3	6,83E-04	8,16E-04	6,83E-04	8,16E-04	1,07832	3,16E-04	2,79E-04	3,01E-03	3,55E-03
4	1,20E-05	1,44E-05	8,32E-06	9,94E-06	1,07212	1,95E-04	1,74E-04	2,74E-04	3,31E-04
5	4,75E-08	5,67E-08	4,75E-08	5,67E-08	1,07803	9,50E-07	8,69E-07	2,94E-05	3,31E-04
6	5,13E-10	6,13E-10	5,93E-10	7,09E-10	1,07374	2,53E-08	2,32E-08	1,52E-07	3,49E-05
7	9,39E-14	1,12E-13	9,39E-14	1,12E-13	1,07702	8,12E-12	8,13E-12	1,15E-08	1,40E-07
8	3,33E-16	4,44E-16	4,44E-16	3,89E-16	1,07456	3,50E-12	3,39E-12	1,82E-12	1,06E-08
9					1,07642	9,33E-14	9,26E-14	1,37E-13	1,87E-12
10					1,07502	3,74E-15	3,83E-15	4,65E-15	1,36E-13

Table 3.8: Example 20 - Performance of both step size strategies with optimal parameters using a quadrature formula of degree 24 with 109 nodes and starting from vector $\mathbf{u}^{(0)} \equiv 1$.

Our experiments prove that our method converges to the solution because the error reaches the machine precision. In the Table A.7 (Appendix A) we can see that $\alpha = 1$ and $2k$ maximized are the best choice for fixed step size strategy. While in Table B.8 (Appendix B) we observe that $\bar{r} = 0.1$ and Start_Iteration $= 4$ are the best choice for adaptive step size strategy. Using these optimal parameters, in Table 3.8 we compare the two step size strategies.

The convergence of fixed step size strategy results much faster than the adaptive step size strategy. Moreover, we note once more that using two different dual functions for fixed step size strategy, we obtain almost the same results. On the other side, in the cases in which we use the adaptive step size strategy we can start the acceleration from the fourth iteration[3], so that we can avoid some unnecessary extrapolation step. In Figure 3.7 we compare the behaviour of the error in the fixed and adaptive step size strategies respectively.

What we want to emphasize is that with our algorithms we can reach the machine precision using a lower numbers of iterations than the ones indicated in [16].

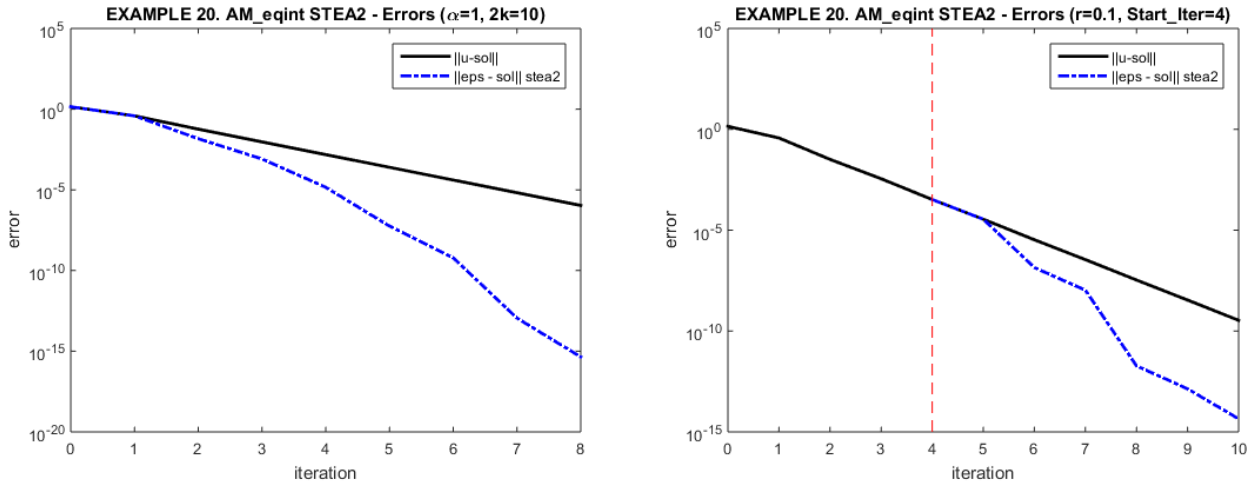

Figure 3.7: Example 20 - Comparison between the two strategies, each of one with the best parameters. Left: fixed step size strategy with $\alpha = 1$ and $2k$ maximized, using $\mathbf{y} \equiv 1$. Right: adaptive step size strategy with $\bar{r} = 0.1$ and Starting_Iteration $= 4$, using $\mathbf{y} = \mathbf{w}$.

[3]We decide to start from the fourth iteration since it coincides with the point from which the α behaviour gets stable.

Example 21

An example where the conditions of applicability of the Theorem 1.3 are satisfied is the linear Fredholm integral equation studied by Alipanah and Esmaeili in [2, p. 5346]:

$$u(t,s) = \frac{1}{5}\int_{[0,\sqrt{\pi}]\times[0,\sqrt{\pi}]} u(x,y)\cos(x\,t)\,\cos(y\,s)\,dxy + 1 - \frac{\sin(\sqrt{\pi}t)\,\sin(\sqrt{\pi}s)}{5\,t\,s}$$

where the solution is $u^*(t) = 1$. In their article they present a method based on interpolation by Gaussian radial basis function. They approximate the integral on Legendre-Gauss-Lobatto nodes and weights for rectangular domain and they obtain a precision of 10^{-8}. Since the Lipschitz condition of kernel is global satisfied, we are sure that our algorithm converges to the unique solution individuated by the authors. In the Table 3.9 we show the results of our algorithm with the two strategies and their optimal parameters using the quadrature formula of degree 18 individuated by [28] and we compare graphically the error in the Figure 3.8.

The previous considerations are repeated here: also in this case it is evident that the behaviour of error is similar. In the adaptive step size strategy we can start the extrapolation from fifth iteration in order to involve the next eight iterations and to obtain a convergence to machine precision. Even if the adaptive strategy is quite better considering the number of iterations involved by extrapolation (hence the memory resources needed in `STEA2` are less than fixed strategy), we have to take into account the computational cost.

	Fixed step size strategy $\boldsymbol{\alpha = 1}$ **Num_Iter = 11 2k=10**				**Adaptive step size strategy** $\boldsymbol{\bar{r} = 0,3}$ **Num_Iter = 15 2k=14 y = w**				
	y ≡ 1		**y = w**			**Start_Iter = 0**		**Start_Iter = 4**	
n	**Residual**	**Error**	**Residual**	**Error**	$\boldsymbol{\alpha^{(n)}}$	**Residual**	**Error**	**Residual**	**Error**
0	9,89E-01	9,80E-01	9,60E-01	9,99E-01		9,64E-01	9,85E-01	9,73E-01	9,95E-01
1	2,30E-01	2,92E-01	2,35E-01	2,97E-01	0,85535	2,70E-01	3,63E-01	3,14E-01	4,31E-01
2	3,51E-02	3,53E-02	4,61E-02	4,42E-02	1,00113	4,90E-02	5,58E-02	8,36E-02	1,20E-01
3	1,49E-02	1,45E-02	1,93E-02	1,84E-02	1,01520	8,84E-04	9,13E-04	2,43E-02	3,51E-02
4	3,01E-04	3,05E-04	8,72E-05	9,10E-05	1,00933	1,14E-04	1,32E-04	7,39E-03	1,07E-02
5	1,09E-05	1,14E-05	6,26E-06	6,58E-06	1,01312	1,07E-05	1,19E-05	2,20E-03	3,18E-03
6	3,96E-07	4,26E-07	6,58E-08	6,93E-08	1,01112	4,89E-07	5,44E-07	6,55E-06	7,66E-06
7	1,33E-08	1,43E-08	2,80E-10	3,06E-10	1,01226	1,28E-09	1,33E-09	1,12E-06	1,31E-06
8	2,20E-11	2,34E-11	4,71E-12	4,98E-12	1,01163	9,99E-11	1,10E-10	3,66E-09	4,08E-09
9	3,61E-13	3,86E-13	8,37E-14	8,84E-14	1,01199	4,23E-11	4,68E-11	2,37E-10	2,64E-10
10	4,44E-15	4,44E-15	2,22E-15	1,78E-15	1,01179	3,26E-13	3,58E-13	8,63E-13	9,53E-13
11					1,01190	3,10E-14	3,45E-14	8,09E-14	8,82E-14
12					1,01184	3,66E-15	3,00E-15	3,22E-15	3,11E-15

Table 3.9: Example 21 - Performances of both step size strategies with optimal parameters using a quadrature formula of degree 24 with 109 nodes and start from vector $\mathbf{u}^{(0)}$ with components randomly decided in $[0, 1]$.

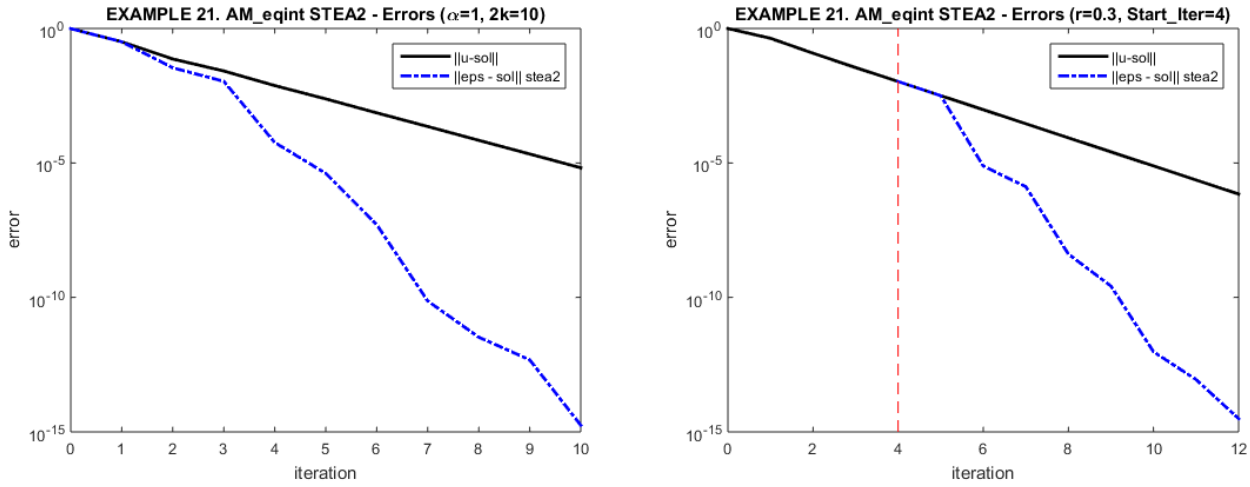

Figure 3.8: Example 21 - Comparison between the two strategies, each of one with the best parameters. Left: fixed step size strategy with $\alpha = 1$ and $2k$ maximized, using $\mathbf{y} \equiv 1$. Right: adaptive step size strategy with $\bar{r} = 0.3$ and Starting_Iteration $= 4$, using $\mathbf{y} = \mathbf{w}$.

Example 22

Finally we want to present an example where the acceleration has an important role. We analyse the Example 1 proposed in [17]. That is the linear Fredholm integral equation

$$u(t,s) = e^{-s-t} - \int_{[0,1]\times[0,1]} s\,t\,e^{(x+y)}\,u(x,y)\,dxy$$

where the authors consider the solution $u^*(t,s) = e^{-t-s} - \frac{ts}{2}$. The authors [17] propose a numerical method based on integral mean value theorem. In particular they prove the convergence of their approach and for this example they obtain a convincing convergence up to the precision 10^{-7} magnitude order.

We establish that the quadrature formula of degree 24 individuated by [28] determines a $\mathcal{E}^{\mathrm{QUAD}}(u^*)$ of machine precision. But we note that there is not a fixed ratio for adaptive step size strategy that provides an interesting or comparable performance to fixed step size strategy. In the Figure 3.9 and in the Table 3.10 we show the behaviour of our algorithm with different strategies, each of one with optimal parameters using the quadrature formula of degree 18 individuated in [28]. Looking at Table A.11 (Appendix A) we get to know that the fixed step size with $\alpha = 1$ gives the same result but the original sequence increases at each iteration. In order to ensure the coherence of the algorithms we have studied, we prefer to set the α parameter so that the original sequence has a decreasing residual. We see that the original sequence converges very slowly while the acceleration given by fixed step size $\alpha = 0.5$ converges to the machine precision in only four iteration.

	Fixed step size strategy $\alpha = 0.5$ **Num_Iter = 11 2k=10**				**Adaptive step size strategy** $\bar{r} = 0,4$ **Num_Iter = 15 2k=14 y = w**				
	y ≡ 1		**y = w**			**Start_Iter = 0**		**Start_Iter = 8**	
n	**Residual**	**Error**	**Residual**	**Error**	$\alpha^{(n)}$	**Residual**	**Error**	**Residual**	**Error**
0	3,69E+00	1,34E+00	3,69E+00	1,34E+00	0,00000	3,69E+00	1,34E+00	3,69E+00	1,34E+00
1	2,19E+00	1,25E+00	5,07E-01	5,07E-01	0,31314	1,06E+00	4,29E-01	1,06E+00	4,29E-01
2	8,92E-02	1,12E-01	2,43E-01	2,43E-01	0,32329	4,91E-01	4,02E-01	4,91E-01	4,02E-01
3	2,86E-02	3,59E-02	1,28E-15	1,22E-15	0,34163	3,07E-01	2,50E-01	3,07E-01	2,50E-01
4	6,66E-16	4,44E-16			0,37527	2,00E-01	1,70E-01	2,00E-01	1,70E-01
5					0,43685	1,71E-01	1,45E-01	1,71E-01	1,45E-01
6					0,53671	1,99E-01	1,69E-01	1,99E-01	1,69E-01
7					0,61327	4,42E-01	3,75E-01	4,42E-01	3,75E-01
8					0,59273	1,71E-01	1,46E-01	7,99E-03	7,98E-03
9					0,60343	3,57E-03	3,33E-03	3,16E-03	3,16E-03
10					0,59824	3,57E-03	2,91E-03	3,11E-05	2,92E-05
11					0,60087	7,29E-04	6,98E-04	6,35E-06	5,96E-06
12					0,59956	3,37E-04	3,22E-04	8,48E-09	8,38E-09
13					0,60022	1,31E-09	1,25E-09	9,25E-10	9,09E-10
14					0,59989	4,64E-11	4,12E-11	1,32E-11	1,27E-11
15					0,60005	6,44E-12	6,31E-12	6,26E-13	6,03E-13
16					0,59997	1,32E-12	1,30E-12	6,16E-15	5,88E-15
17					0,60000	1,32E-12	1,30E-12	4,39E-15	4,39E-15

Table 3.10: Example 22 - Performances of both step size strategies with optimal parameters using a quadrature formula of degree 18 with 64 nodes and start from vector $\mathbf{u}^{(0)} \equiv 1$.

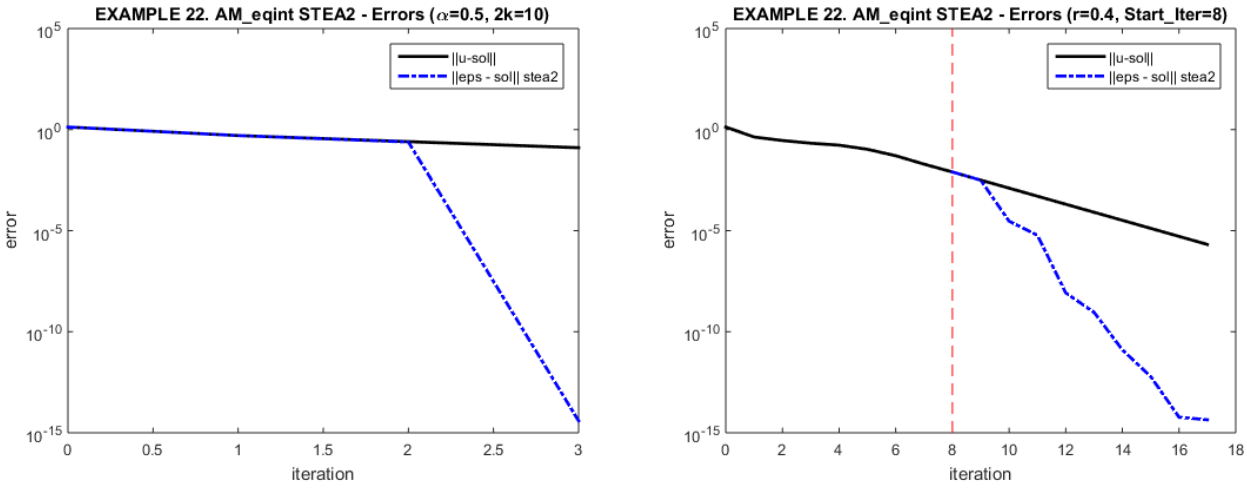

Figure 3.9: Example 22 - Comparison between the two strategies, each of one with the best parameters. Left: fixed step size strategy with $\alpha = 0.5$ and $2k$ maximized, using $\mathbf{y} \equiv 1$. Right: adaptive step size strategy with $\bar{r} = 0.4$ and Starting_Iteration $= 8$, using $\mathbf{y} = \mathbf{w}$.

3.2 Examples in disk

In this section we present two examples where the domain is the disk having the center in the origin, and a radius equal to 1,

$$D = \left\{ (x, y) \in \mathbb{R}^2 \middle| x^2 + y^2 \leq 1 \right\}$$

We built both because we did not find any article which has ever discussed an integral equation of second kind defined on this domain. We observe that both examples satisfy the condition of Theorem 1.3 because they satisfy the Lipshitz condition in all domain with a Lipshitz's constant less than $\frac{1}{\pi}$.

But the real problem with this domain is the difficulty to find a quadrature formula with high degree. So for this domain a precision near to the machine's one will not be easy to reach, as we will see experimentally in our examples. Since it is not our aim to deepen the quadrature formulas, we find a good compromise in the use of the formulas identified by Kim and Song [22]. We use their formula of degree 15 truncate on 15-th digits because above this degree Conedera establishes in his thesis that presumably the truncation determines not negligible errors[4]. This formula have 44 nodes and positive weight and we show through the Figure 3.10 the distribution of the nodes.

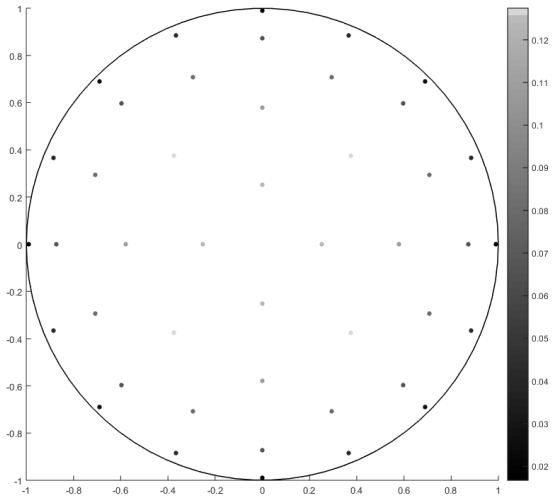

Figure 3.10: The nodes and weights of quadrature formula that we use for Example 23 and Example 24 defined on disk D.

[4]See Conedera, *Formule di cubatura minimali per il disco*, 2015 for more details.

Example 23

The first example that we want to explore is the linear integral equation given by

$$u(t,s) = \frac{1}{2}\int_D u(x,t)\,\frac{x\,t^2}{x^2+y^2}\,dxy + t^2\frac{32-\pi}{\pi} + t\,s^2$$

where the solution is $u^*(t,s) = t\,s^2 + t^2$. We establish that for this quadrature formula we have $\mathcal{E}^{\mathrm{QUAD}}(u^*) \sim 8\cdot 10^{-4}$. Since we have global Lipshitz property we employ the simple Picard Iteration Method (i.e. `RPIM` with $\alpha \equiv 1$) and we observe in Table 3.11 that it converges so faster (in only four iterations) and it reach the magnitude order of $\mathcal{E}^{\mathrm{QUAD}}(u^*)$. In the same table we show the results obtained by both strategies, each of one with optimal parameters and we note that the convergence of accelerated sequence accelerated is worse than the original sequence generated by the simple Picard iteration. We deduce that, for this particular case where the original sequence converges fast, extrapolation does not provide a good improvement. And in Figure 3.11 we show the residuals and the errors for the adaptive step size strategies for the fixed ratio $\bar{r} = 0.6$ and we can see the stagnation of error due to the low order of $\mathcal{E}^{\mathrm{QUAD}}(u^*)$. This example is interesting because the convergence of simple Picard Iteration Method in Chapter 1 are sufficient to identify an excellent iterative method.

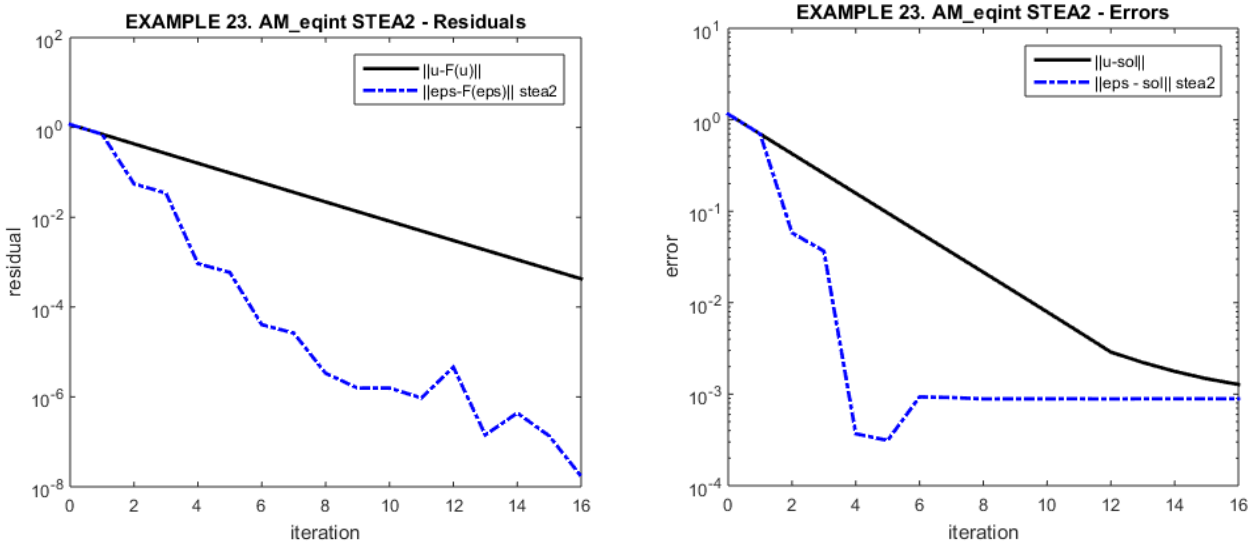

Figure 3.11: Example 23 - The stagnation of error due to error of the quadrature formula using the adaptive step size strategy with $\bar{r} = 0.6$, Start_Iteration $= 0$, $2k = 16$ and $\mathbf{y} = \mathbf{w}$.

	RPIM $\alpha = 1$		**Fixed step size strategy** $\alpha = 1$ **Num_Iter = 11 2k=10** $\mathbf{y} \equiv 1$		**Adaptive step size strategy ratio = 0,6 Num_Iter = 17 2k=16** $y = w$ **Start_Iter=0**	
n	**Residual**	**Error**	**Residual**	**Error**	**Residual**	**Error**
0	1,16E+00	1,15E+00	1,16E+00	1,15E+00	1,16E+00	1,15E+00
1	9,53E-02	9,62E-02	9,53E-02	9,62E-02	7,08E-01	7,00E-01
2	8,88E-16	8,92E-04	2,90E-03	3,79E-03	5,54E-02	5,81E-02
3	2,22E-16	8,92E-04	2,22E-16	8,92E-04	3,46E-02	3,67E-02
4					9,30E-04	3,69E-04
5					5,91E-04	3,13E-04
6					4,06E-05	9,34E-04
7					2,64E-05	9,19E-04
8					3,36E-06	8,88E-04
9					1,57E-06	8,90E-04
10					1,58E-06	8,90E-04
11					9,47E-07	8,91E-04
12					4,62E-06	8,87E-04
13					1,42E-07	8,92E-04
14					4,39E-07	8,91E-04
15					1,36E-07	8,92E-04
16					1,71E-08	8,92E-04

Table 3.11: Example 23 - Performance of the Picard Iteration method compared to both step size strategies start from vector $\mathbf{u}^{(0)} \equiv 1$.

Example 24

Now we study a nonlinear integral equation also defined on a disk.
We consider

$$u(t,s) = \frac{e}{\pi(e-2)} \int_D (t^2 + u(x,y)\, s^2)\, e^{-u(x,y)}\, dxy - \frac{s^2}{e-2}$$

where the solution is $u^*(t,s) = t^2 + s^2$. Using the quadrature formula on the disk defined before we obtain $\mathcal{E}^{\mathrm{QUAD}}(u^*) \sim 6 \cdot 10^{-9}$. Therefore we know that our algorithm do not reach the machine precision but the magnitude order of $\mathcal{E}^{\mathrm{QUAD}}(u^*) \sim 6{\cdot}10^{-9}$ (only if it converges). In this case the nonlinearity determines a slow convergence of simple Picard Iteration Method as we can see in Figure 3.12 (left). In the same figure we note that we obtain a good acceleration by extrapolation. In this case too, we have as optimal parameters of fixed step size strategy the value $\alpha = 1$ (i.e. using the simple Picard iteration method) and $2k$ maximized, as we can see in Table A.15 (Appendix A). While in Table B.15 (Appendix B) we determine that the optimal parameter for the adaptive strategy is $\bar{r} = 0.2$. In the Figure 3.12 (right) we can observe that the extrapolation is close to the sequence obtained with the adaptive strategy. In the Table 3.12 we report the numerical result for these parameters. This example confirm the feelings collected so far: the fixed step size strategy is more competitive respect to adaptive strategy in all condition that we have studied.

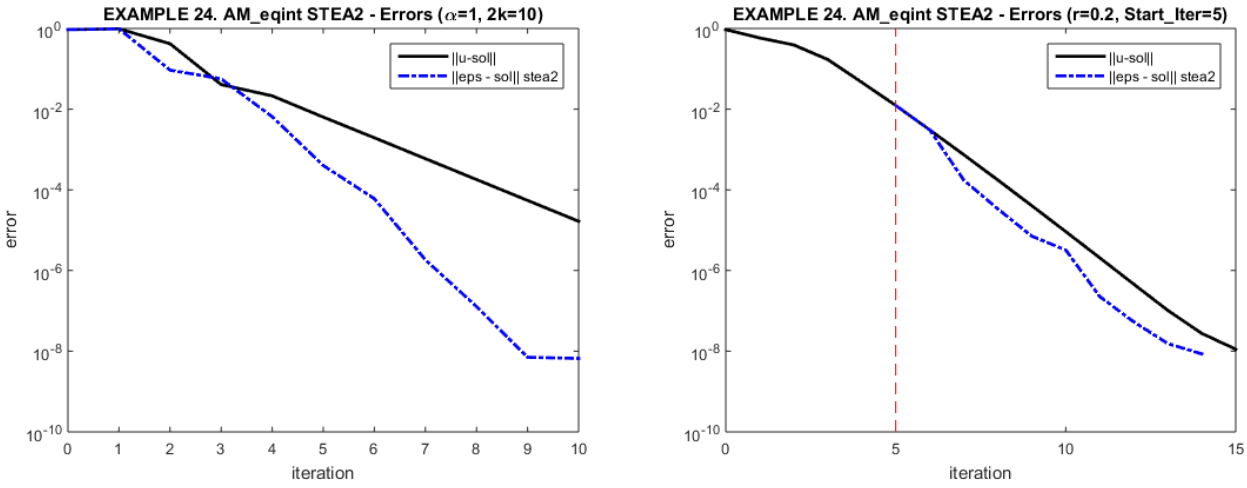

Figure 3.12: Example 24 - Comparison between the two strategies, each of one with the best parameters. Left: fixed step size strategy with $\alpha = 1$ and $2k$ maximized, using $\mathbf{y} \equiv 1$. Right: adaptive step size strategy with $\bar{r} = 0.2$ and Starting_Iteration $= 5$, using $\mathbf{y} = \mathbf{w}$.

	Fixed step size strategy $\alpha = 1$ **Num_Iter = 11 2k=10**				**Adaptive step size strategy** $\bar{r} = 0,2$ **Num_Iter = 15 2k=14** $\mathbf{y} = w$				
	$\mathbf{y} \equiv 1$		$\mathbf{y} = w$			**Start_Iter = 0**		**Start_Iter = 5**	
n	**Residual**	**Error**	**Residual**	**Error**	$\alpha^{(n)}$	**Residual**	**Error**	**Residual**	**Error**
0	1,00E+00	9,36E-01	1,00E+00	9,36E-01		1,00E+00	9,36E-01	1,00E+00	9,36E-01
1	1,40E+00	9,80E-01	1,40E+00	9,80E-01	0,60245	3,75E-01	5,82E-01	3,75E-01	5,82E-01
2	7,31E-02	9,26E-02	1,37E-01	1,47E-01	0,63838	3,20E-01	3,39E-01	3,10E-01	3,90E-01
3	5,37E-02	5,65E-02	5,37E-02	5,65E-02	0,71162	1,03E-01	1,07E-01	1,47E-01	1,70E-01
4	6,15E-03	6,49E-03	4,75E-02	4,93E-02	0,84112	2,25E-02	2,74E-02	4,19E-02	4,62E-02
5	3,89E-04	4,01E-04	3,89E-04	4,01E-04	0,80857	9,74E-03	8,61E-03	1,15E-02	1,23E-02
6	5,84E-05	6,02E-05	4,40E-05	4,54E-05	0,81031	2,87E-03	2,61E-03	2,88E-03	3,05E-03
7	1,78E-06	1,83E-06	1,78E-06	1,83E-06	0,80179	1,93E-04	1,93E-04	1,74E-04	1,74E-04
8	1,30E-07	1,28E-07	1,13E-07	1,10E-07	0,80117	4,07E-05	4,07E-05	3,38E-05	3,37E-05
9	4,33E-10	7,07E-09	4,33E-10	7,07E-09	0,80022	4,69E-06	4,69E-06	7,11E-06	7,12E-06
10	3,95E-12	6,63E-09	4,16E-12	6,63E-09	0,80009	1,19E-06	1,20E-06	3,21E-06	3,22E-06
11	1,70E-13	6,62E-09	1,70E-13	6,62E-09	0,80002	3,15E-07	3,21E-07	2,19E-07	2,26E-07
12	8,88E-15	6,62E-09	9,44E-15	6,62E-09	0,80001	4,30E-08	4,96E-08	4,69E-08	5,35E-08
13	3,68E-15	6,62E-09	1,89E-15	6,62E-09	0,80000	1,03E-08	1,69E-08	8,84E-09	1,55E-08
14					0,80000	2,34E-09	8,96E-09	1,96E-09	8,58E-09

Table 3.12: Example 24 - Performances of both step size strategies with optimal parameters using a quadrature formula described before and start from vector $\mathbf{u}^{(0)} \equiv 1$.

Conclusion

In the first chapter we study the theoretical aspects involved when we solve the Urysohn integral equations. The choice of the quadrature formula and of the starting point are important for converging to the solution, as we have seen in the Example 2. In fact, there are conditions within which the problem appears to be a contraction, as we have seen in Theorem 1.3, and we have to start in the region described by them in order to ensure the convergence of the simple Picard Iteration method. Hence a relaxation of this method is not necessary and when we apply the extrapolation on the sequence given by Relaxed Picard Iteration method for the fixed strategy and our adaptive strategy, we observe that the best performance is obtained by the extrapolation of the simple Picard Iteration. Hence if a integral equation satisfies the conditions of the Theorem 1.3, the application of the extrapolation with $2k$ maximized (and using any dual vector) on the fixed step size strategy with $\alpha = 1$ (i.e. the simple Picard Iteration) is effective compared to what observed in the original Brezinski, Redivo-Zaglia's work [10] and to the other methods that we have explored in this thesis.

We have also studied some integral equations where these conditions are not satisfied: these cases are frequent in application. In literature a fixed point for a general non-contraction operator can be determined by Relaxed Picard Iteration method for some values of α or for an opportune adaptive strategy. The performance obtained by the fixed step size strategy is similar to that of our adaptive strategy when we compare both strategies, each of one with the best parameters.

In both cases, the two strategies offer a similar performance but we have seen that the adaptive strategy is sensitive[5] to the choice of $\mathbf{y}$ and it can be expensive in the determination of the appropriate step size to each iteration. In Table 3.13 we compare the computation times[6] between both strategies for all examples that we presented in this thesis (each of one with the best parameters). We observe that the extrapolation applied to the fixed step size strategy is more competitive

[5]We can see that from the numerous absent values in the tables of Appendix B in which we have chosen $\mathbf{y} \equiv 1$.

[6]We carry out the computations with an Intel(R) Core(TM) i3-4012Y CPU @ 1.50 GHz, 4.00 GB, 64 bit processor.

to our adaptive strategy (extrapolated or not) but also offers two small significant positive aspects that we intend to mention. $\bar{r}$ is the only parameter of the method and it is intuitive for users compared to a good choice of parameter α (which is necessary for the fixed step size strategy). Another positive aspect is that the behaviour of the error is regular and we usually use less iterations of the extrapolation method. It is also necessary to point out that in these situations we have never encountered the presence of singularities to be managed. Hence the implementation of Cordellier's particular rule has not been decisive for the study of these strategies in their optimal settings.

	Fixed step size strategy	**Adaptive step size strategy**
Example 2	0,072960 s	0,092894 s
Example 3	0,198796 s	0,490890 s
Example 4	0,058136 s	0,056227 s
Example 20	0,940132 s	2,869472 s
Example 21	0,364884 s	0,816837 s
Example 22	0,865686 s	0,109357 s
Example 23	0,142959 s	0,633196 s
Example 24	0,210312 s	0,716582 s

Table 3.13: Computation time of both strategies for all examples, each of one with the best parameters.

For these reasons, we argue that the fixed step size strategy proposed by Brezinski and Redivo-Zaglia [10] is preferred over our adaptive strategy. In our opinion a study of a different adaptive strategy can lead to a better performance that the one obtained using the fixed step size strategy. In fact we adapt the step size in order to get close to the kernel of the Shanks' extrapolation along the direction $\mathbf{y}$ only. An idea for a development of this thesis is the use of the topological extrapolation with adaptive dual vector $\mathbf{y}$ in order to involve better directions from which to obtain interesting performances.

The critical aspect that we can highlight is the cost in determining the effective parameters in the cases in which the conditions of this theorem are not satisfied. Obviously, it will be necessary to evaluate a set of possible choices of α, but we believe it is possible to eliminate the need to test each of them for different $2k$ values. In fact we have seen the correlation between ratio of the residual and the convergence in the last section of Chapter 2. An idea that we think can resolve this aspect is to take the maximum value of $2k$ so that we get the entire ε-array. If we notice a stabilization of the ratio of the residual after the m-th iterations we consider as extrapolated vector the $\widetilde{\boldsymbol{\varepsilon}}_{2k-m}^{(n)}$ at n-th iteration, instead of taking the tip of complete ε-array, $\widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n)}$. In other words the idea is to adaptively modify the parameter $2k$ based on the considerations that we applied in our adaptive step size strategy. This idea could be the subject of new studies because it resembles to be promising to improve the flexibility for users of this numerical method.

Appendix A

Reference tables for fixed step size strategy

In the following tables we report the number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector for every combination of α (vertical) $2k$ (horizontal). These is important to establish the optimal parameter for the fixed step size strategy that we discuss before.

When the value is not reported it means that the method has not produced a result or the previous one had already led to the best solution achievable for that column. We have computed the values reported in these tables by `comparison_for_fixed_strategy.m` contained in the software pack that we wrote for thesis. For our experiments the tolerance parameter, Tol, is set to $5 \cdot 10^{-15}$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2K=2	50	5,98E-05	6,60E-05	50	4,50E-08	4,91E-08	50	6,86E-13	7,41E-13	40	2,94E-15	3,11E-15	28	1,42E-15	1,47E-15
2K=4	50	4,85E-06	4,87E-06	50	1,01E-12	1,00E-12	43	1,11E-15	1,17E-15	28	1,03E-15	8,60E-16	18	1,53E-15	1,64E-15
2K=6	50	5,95E-09	4,87E-09	50	8,56E-13	8,65E-13	41	9,44E-16	7,77E-16	29	2,83E-15	3,05E-15	17	3,44E-15	3,77E-15
2K=8	50	7,68E-09	6,55E-09	50	9,23E-13	9,48E-13	42	1,58E-15	1,75E-15	24	1,55E-15	1,19E-15	16	3,36E-15	3,69E-15
2K=10	50	1,07E-08	9,53E-09	50	6,45E-14	5,16E-14	38	3,39E-15	3,52E-15	22	2,66E-15	2,80E-15	16	1,67E-16	1,11E-16
2K=12	50	1,08E-08	9,57E-09	50	2,65E-11	2,74E-11	38	1,78E-15	1,44E-15	21	2,50E-15	2,75E-15	16	1,67E-16	1,11E-16
2K=14	50	2,69E-09	2,37E-09	50	7,66E-13	7,91E-13	32	9,99E-16	8,60E-16	21	2,50E-15	2,75E-15	16	1,67E-16	1,11E-16
2K=16	50	9,53E-10	8,17E-10	50	1,05E-12	1,11E-12	31	1,14E-15	1,25E-15	18	1,35E-11	1,48E-11	16	1,67E-16	1,11E-16
2K=18	50	6,51E-09	5,77E-09	48	1,61E-15	1,69E-15	28	1,22E-15	8,33E-16	18	1,35E-11	1,48E-11			
2K=20	50	3,98E-09	3,54E-09	50	1,12E-14	1,23E-14	31	1,05E-15	1,08E-15						
2K=22	50	2,61E-10	2,38E-10	23	3,38E-09	3,07E-09	32	8,33E-16	8,05E-16						
2K=24	50	2,93E-10	2,53E-10	23	3,38E-09	3,07E-09	30	3,33E-15	3,58E-15						
2K=26	50	6,29E-09	5,57E-09				30	3,97E-15	4,30E-15						
2K=28	50	9,23E-10	8,30E-10				34	1,45E-14	1,54E-14						
2K=30	50	5,17E-10	4,67E-10				32	4,14E-15	4,50E-15						
2K=32	50	9,12E-10	8,24E-10				32	4,14E-15	4,50E-15						
2K=34	50	2,58E-09	2,31E-09												
2K=36	48	3,56E-10	2,65E-10												
2K=38	48	2,05E-08	2,04E-08												
2K=40	48	2,18E-10	1,75E-10												
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2K=2	19	1,03E-15	1,11E-15	15	8,33E-16	8,33E-16	20	4,47E-15	4,19E-15	17	1,14E-15	8,05E-16	21	3,52E-15	2,39E-15
2K=4	17	4,02E-15	4,27E-15	15	5,27E-16	5,27E-16	14	3,61E-16	3,05E-16	14	4,77E-15	3,72E-15	15	1,89E-15	1,22E-15
2K=6	14	2,00E-15	2,05E-15	12	3,61E-16	3,61E-16	12	4,16E-16	3,89E-16	14	1,28E-15	1,03E-15	12	2,97E-15	2,00E-15
2K=8	13	8,33E-16	9,16E-16	12	1,67E-16	1,94E-16	11	3,47E-15	3,33E-15	13	1,67E-16	1,39E-16	11	1,78E-15	1,17E-15
2K=10	13	1,80E-15	1,92E-15	12	1,67E-16	1,94E-16	12	1,67E-16	1,67E-16	13	1,11E-16	8,33E-17	11	1,78E-15	1,17E-15
2K=12	14	5,00E-16	5,00E-16	12	1,67E-16	1,94E-16	12	1,67E-16	1,67E-16	13	1,11E-16	8,33E-17	11	1,78E-15	1,17E-15
2K=14	14	5,00E-16	5,00E-16							13	1,11E-16	8,33E-17			
2K=16															

Table A.1: Example 2 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2K=2	50	8,00E-05	8,19E-05	50	1,31E-07	1,34E-07	50	2,88E-12	2,94E-12	41	9,99E-16	1,08E-15	29	1,89E-15	1,89E-15
2K=4	50	2,63E-05	2,70E-05	50	4,32E-12	4,44E-12	50	5,70E-14	5,80E-14	29	3,80E-15	3,86E-15	19	1,69E-15	1,67E-15
2K=6	50	1,97E-08	2,04E-08	50	4,09E-12	4,20E-12	43	2,89E-15	2,91E-15	29	1,76E-14	1,79E-14	18	3,61E-16	3,89E-16
2K=8	50	2,46E-08	2,55E-08	50	3,03E-13	3,11E-13	44	9,99E-16	9,71E-16	27	1,94E-15	2,00E-15	19	1,69E-15	1,67E-15
2K=10	50	4,05E-08	4,19E-08	50	1,48E-12	1,52E-12	45	3,41E-15	3,47E-15	22	4,77E-15	4,94E-15	18	9,71E-16	1,03E-15
2K=12	50	4,10E-08	4,23E-08	50	1,44E-12	1,48E-12	31	4,08E-15	4,02E-15	20	3,19E-15	3,19E-15	17	3,86E-15	3,91E-15
2K=14	50	2,10E-09	2,18E-09	50	1,44E-12	1,48E-12	23	4,47E-15	3,75E-15	22	8,60E-16	8,88E-16	18	6,11E-16	5,83E-16
2K=16	50	4,02E-09	4,14E-09	50	6,51E-13	6,68E-13	46	3,44E-15	3,47E-15	18	5,86E-11	6,00E-11	18	1,67E-16	1,39E-16
2K=18	50	2,33E-08	2,40E-08	50	1,35E-13	1,39E-13	36	3,39E-15	3,64E-15	18	5,86E-11	6,00E-11	18	1,67E-16	1,39E-16
2K=20	50	9,59E-09	9,91E-09	50	3,55E-14	3,63E-14	34	2,78E-15	2,83E-15						
2K=22	39	2,38E-06	2,46E-06	23	7,83E-08	8,09E-08	40	4,16E-15	3,19E-15						
2K=24	39	2,39E-07	2,47E-07	23	7,83E-08	8,09E-08	29	3,66E-15	3,66E-15						
2K=26	39	1,24E-06	1,28E-06				34	3,08E-15	3,16E-15						
2K=28	39	1,96E-06	2,03E-06				34	1,44E-15	1,47E-15						
2K=30	39	1,56E-06	1,61E-06				34	4,25E-15	4,33E-15						
2K=32	39	4,61E-06	4,77E-06				36	2,44E-15	2,58E-15						
2K=34	39	4,69E-06	4,85E-06				37	4,69E-15	4,80E-15						
2K=36	39	1,36E-06	1,40E-06				46	1,83E-13	1,90E-13						
2K=38	38	9,19E-07	9,51E-07				46	2,17E-13	2,24E-13						
2K=40							46	3,58E-14	3,71E-14						
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2K=2	19	3,66E-15	3,72E-15	15	2,50E-15	2,47E-15	20	3,89E-15	3,80E-15	17	1,14E-15	8,05E-16	21	3,52E-15	2,39E-15
2K=4	18	2,41E-15	2,50E-15	15	1,39E-15	1,42E-15	14	9,71E-16	9,71E-16	14	2,83E-15	3,05E-15	15	1,89E-15	1,22E-15
2K=6	14	4,08E-15	4,16E-15	12	8,05E-16	8,05E-16	12	6,94E-16	6,94E-16	14	1,94E-15	2,08E-15	12	2,97E-15	2,03E-15
2K=8	13	1,44E-15	1,47E-15	12	8,33E-17	1,11E-16	12	8,33E-17	8,33E-17	13	3,89E-16	4,16E-16	11	1,67E-15	1,08E-15
2K=10	14	1,69E-15	1,69E-15	12	8,33E-17	1,11E-16	12	8,33E-17	8,33E-17	13	2,22E-16	2,50E-16	11	1,67E-15	1,08E-15
2K=12	14	1,69E-15	1,69E-15	12	8,33E-17	1,11E-16	12	8,33E-17	8,33E-17	13	2,22E-16	2,50E-16	11	1,67E-15	1,08E-15
2K=14	14	1,69E-15	1,69E-15							13	2,22E-16	2,50E-16			

Table A.2: Example 2 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2K=2	50	8,40E-05	8,29E-05	50	3,82E-09	3,90E-09	50	5,54E-14	5,51E-14	39	4,00E-15	3,33E-15	50	3,51E-13	3,62E-13
2K=4	50	5,29E-07	4,77E-07	50	3,91E-13	3,79E-13	46	3,20E-15	3,33E-15	39	4,00E-15	3,33E-15	31	2,55E-15	2,33E-15
2K=6	50	6,95E-08	5,31E-08	50	1,68E-13	1,30E-13	39	3,40E-15	3,33E-15	31	2,87E-15	6,06E-15	32	9,02E-16	2,00E-15
2K=8	50	8,05E-09	6,11E-09	50	7,96E-13	7,56E-13	39	1,26E-15	3,76E-15	25	2,15E-15	1,89E-15	23	1,55E-15	2,00E-15
2K=10	50	7,60E-10	4,71E-10	50	6,71E-14	6,10E-14	28	5,00E-15	5,44E-15	21	6,66E-16	2,21E-15	24	6,94E-16	3,11E-15
2K=12	50	4,58E-09	3,41E-09	50	2,95E-14	3,40E-14	26	3,55E-15	3,59E-15	23	1,75E-15	4,00E-15	21	3,99E-15	2,22E-15
2K=14	50	2,96E-10	2,58E-10	50	3,41E-14	3,45E-14	26	4,18E-15	6,76E-15	24	3,00E-15	3,11E-15	23	2,22E-16	2,67E-15
2K=16	50	2,74E-09	2,09E-09	50	4,36E-14	4,52E-14	32	1,33E-15	2,25E-15	24	2,66E-15	2,89E-15	19	4,00E-15	3,33E-15
2K=18	50	1,20E-10	9,84E-11	50	4,88E-15	3,33E-15	23	Inf	7,64E+07	20	2,54E-13	2,65E-13	20	2,36E-15	2,33E-15
2K=20	50	1,76E-10	1,42E-10	50	1,72E-14	2,37E-14				20	2,54E-13	2,65E-13	20	2,36E-15	2,33E-15
2K=22	50	1,77E-10	1,27E-10	50	1,33E-14	1,83E-14									
2K=24	50	4,03E-10	3,15E-10	50	4,22E-15	3,77E-15									
2K=26	50	9,99E-11	7,58E-11	50	2,17E-14	2,45E-14									
2K=28	50	2,27E-11	1,65E-11	50	1,58E-14	1,36E-14									
2K=30	50	6,45E-10	4,88E-10	50	1,58E-14	1,36E-14									
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2K=2	50	1,90E-07	5,94E-08	50	9,72E-01	3,36E-01	50	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	50	1,18E+01	9,15E+00
2K=4	50	3,35E-10	1,05E-10	50	9,72E-01	3,36E-01	50	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	50	1,18E+01	9,15E+00
2K=6	50	2,22E-13	6,95E-14	50	9,72E-01	3,36E-01	50	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	50	1,16E+01	8,97E+00
2K=8	46	4,88E-15	5,16E-15	48	9,72E-01	3,36E-01	47	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	50	1,16E+01	8,97E+00
2K=10	49	3,55E-15	3,56E-15	48	9,72E-01	3,36E-01	47	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	50	1,16E+01	8,97E+00
2K=12	40	4,22E-15	5,56E-15	48	9,72E-01	3,36E-01	35	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=14	47	2,45E-14	9,21E-15	43	9,72E-01	3,36E-01	35	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=16	39	8,33E-16	1,89E-15	43	9,72E-01	3,36E-01	35	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=18	40	3,11E-15	1,50E-15	43	9,72E-01	3,36E-01	35	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=20	38	1,73E-15	1,78E-15	43	9,72E-01	3,36E-01	35	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00			
2K=22	37	3,44E-15	3,89E-15	43	9,72E-01	3,36E-01	35	2,68E+00	1,12E+00	49	4,84E+00	2,55E+00			
2K=24	37	4,22E-15	2,89E-15	43	9,72E-01	3,36E-01	30	2,68E+00	1,12E+00	49	4,84E+00	2,55E+00			
2K=26	27	5,59E-12	4,88E-12	43	9,72E-01	3,36E-01	30	2,68E+00	1,12E+00	49	4,84E+00	2,55E+00			
2K=28	27	5,59E-12	4,88E-12	41	9,72E-01	3,36E-01	30	2,68E+00	1,12E+00	49	4,84E+00	2,55E+00			
2K=30				34	9,72E-01	3,36E-01	30	2,68E+00	1,12E+00	49	4,84E+00	2,55E+00			

Table A.3: Example 3 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2K=2	50	8,40E-05	8,29E-05	50	3,82E-09	3,90E-09	50	5,84E-14	6,01E-14	39	3,33E-15	2,89E-15	50	3,51E-13	3,62E-13
2K=4	50	5,29E-07	4,77E-07	50	3,73E-13	3,72E-13	50	3,00E-14	3,22E-14	35	3,65E-15	3,44E-15	31	2,36E-15	2,11E-15
2K=6	50	6,93E-08	5,30E-08	50	2,54E-13	2,35E-13	37	1,39E-15	4,22E-15	26	4,44E-15	3,33E-15	32	2,00E-15	3,24E-15
2K=8	50	8,30E-09	6,24E-09	50	1,42E-12	1,29E-12	31	2,22E-15	4,14E-15	28	3,66E-15	3,11E-15	23	1,11E-15	1,55E-15
2K=10	50	2,84E-09	2,37E-09	50	9,87E-14	1,06E-13	31	4,11E-15	3,22E-15	26	1,11E-15	1,78E-15	24	3,12E-15	5,53E-15
2K=12	50	6,68E-09	5,08E-09	40	2,55E-15	3,22E-15	31	2,88E-15	5,43E-15	23	4,19E-15	7,13E-15	21	2,94E-15	6,10E-15
2K=14	50	6,65E-09	5,05E-09	45	4,52E-15	2,22E-15	32	2,78E-15	3,00E-15	24	1,41E-15	2,11E-15	22	6,66E-16	2,16E-15
2K=16	50	8,15E-09	6,18E-09	50	1,37E-12	1,70E-12	29	2,08E-15	2,11E-15	24	6,94E-16	1,95E-15	19	4,00E-15	3,11E-15
2K=18	50	1,04E-08	7,99E-09	50	1,68E-14	1,79E-14	23	Inf	2,79E+09	20	2,38E-13	2,51E-13	20	2,51E-15	2,55E-15
2K=20	50	4,22E-09	3,44E-09	49	4,15E-15	6,81E-15				20	2,38E-13	2,51E-13	20	2,51E-15	2,55E-15
2K=22	50	3,41E-09	2,13E-09	50	1,05E-14	1,48E-14									
2K=24	50	2,67E-07	1,88E-07	50	6,66E-15	6,44E-15									
2K=26	50	2,17E-07	1,52E-07	50	4,35E-15	4,66E-15									
2K=28	43	1,62E-08	1,20E-08	44	4,29E-15	1,89E-15									
2K=30	43	2,73E-10	2,74E-10	48	5,00E-15	6,88E-15									
2K=32	43	1,25E-09	8,62E-10	50	6,47E-15	1,02E-14									
2K=34	43	3,12E-09	2,23E-09	44	3,33E-15	2,11E-15									
2K=36	43	5,10E-10	3,42E-10	44	3,22E-15	2,69E-15									
2K=38	43	5,17E-10	3,47E-10	50	7,68E-14	6,49E-14									
2K=40	43	1,33E-09	9,35E-10	49	Inf	4,11E+13									
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2K=2	50	1,90E-07	5,94E-08	50	9,72E-01	3,36E-01	50	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	50	1,18E+01	9,15E+00
2K=4	50	3,35E-10	1,05E-10	50	9,72E-01	3,36E-01	50	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=6	50	2,22E-13	6,94E-14	50	9,72E-01	3,36E-01	50	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=8	47	1,78E-15	3,66E-15	50	9,72E-01	3,36E-01	42	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=10	47	1,78E-15	3,60E-15	50	9,72E-01	3,36E-01	42	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=12	39	4,35E-15	6,21E-15	47	9,72E-01	3,36E-01	42	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=14	43	3,55E-15	1,44E-15	47	9,72E-01	3,36E-01	42	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=16	44	1,44E-15	2,63E-15	43	9,72E-01	3,36E-01	42	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=18	41	2,44E-15	1,77E-15	43	9,72E-01	3,36E-01	37	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00	17	1,18E+01	9,15E+00
2K=20	38	2,11E-15	2,66E-15	43	9,72E-01	3,36E-01	28	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00			
2K=22	38	3,77E-15	3,55E-15	43	9,72E-01	3,36E-01	28	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00			
2K=24	38	4,66E-15	3,77E-15	43	9,72E-01	3,36E-01	28	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00			
2K=26	27	5,57E-12	4,88E-12	43	9,72E-01	3,36E-01	28	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00			
2K=28	27	5,57E-12	4,88E-12	43	9,72E-01	3,36E-01	28	2,68E+00	1,12E+00	50	4,84E+00	2,55E+00			
2K=30				34	9,72E-01	3,36E-01				50	4,84E+00	2,55E+00			
2K=32				34	9,72E-01	3,36E-01				34	4,84E+00	2,55E+00			
2K=34				34	9,72E-01	3,36E-01				34	4,84E+00	2,55E+00			
2K=36															

Table A.4: Example 3 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2K=2	50	1,46E-03	1,58E-03	50	5,65E-06	6,10E-06	50	9,92E-09	1,07E-08	50	6,30E-12	6,81E-12	48	4,88E-15	5,77E-15
2K=4	50	2,90E-04	3,01E-04	50	3,50E-08	3,66E-08	50	4,76E-13	4,89E-13	50	3,18E-13	3,44E-13	42	4,88E-15	5,77E-15
2K=6	50	4,11E-06	4,20E-06	50	3,21E-10	3,36E-10	50	1,58E-12	1,70E-12	50	3,46E-14	3,73E-14	41	2,66E-15	2,66E-15
2K=8	50	2,59E-06	2,66E-06	50	1,07E-10	1,13E-10	50	1,64E-12	1,76E-12	50	7,24E-14	7,86E-14	40	3,55E-15	3,11E-15
2K=10	50	1,84E-06	1,89E-06	50	2,55E-12	2,79E-12	50	2,26E-13	2,44E-13	44	2,22E-15	2,66E-15	30	2,22E-15	1,78E-15
2K=12	50	2,13E-07	2,19E-07	50	1,23E-11	1,23E-11	50	6,30E-13	6,82E-13	36	3,55E-15	3,11E-15	35	4,00E-15	4,00E-15
2K=14	50	1,35E-07	1,39E-07	50	2,67E-11	2,81E-11	39	1,00E-12	1,04E-12	38	1,78E-14	1,91E-14	34	3,55E-15	3,11E-15
2K=16	50	1,51E-07	1,56E-07	50	1,84E-11	1,90E-11	39	2,14E-12	2,22E-12	38	8,88E-15	9,33E-15	28	4,00E-15	4,88E-15
2K=18	50	1,17E-07	1,20E-07	50	2,14E-11	2,20E-11	39	1,49E-13	1,50E-13	37	1,02E-14	1,11E-14	31	5,82E-14	5,91E-14
2K=20	50	4,97E-08	5,12E-08	50	2,18E-11	2,23E-11	39	5,25E-12	5,52E-12	37	4,06E-13	4,43E-13	31	5,82E-14	5,91E-14
2K=22	50	5,69E-07	5,85E-07	50	2,73E-12	2,73E-12	39	1,03E-12	1,08E-12	37	1,28E-13	1,41E-13	31	1,44E-13	1,54E-13
2K=24	50	9,96E-09	9,94E-09	50	9,79E-12	9,96E-12	39	9,98E-13	1,06E-12	31	3,55E-15	4,88E-15	31	9,77E-15	1,11E-14
2K=26	50	9,15E-09	9,14E-09	50	3,48E-11	3,61E-11	39	6,87E-13	7,14E-13	30	1,07E-14	1,11E-14	31	9,77E-15	1,11E-14
2K=28	50	8,96E-09	8,94E-09	50	2,24E-11	2,31E-11	39	1,30E-13	1,41E-13	30	4,70E-13	5,09E-13	31	1,29E-14	1,51E-14
2K=30	50	8,93E-09	8,92E-09	50	1,18E-11	1,23E-11	39	9,26E-13	9,79E-13	30	4,70E-13	5,09E-13	31	3,11E-15	3,11E-15
2K=32	50	9,84E-09	9,82E-09	50	1,26E-11	1,31E-11	39	3,55E-13	3,83E-13				31	3,11E-15	3,11E-15
2K=34	50	1,11E-08	1,11E-08	50	8,03E-12	8,19E-12	39	1,07E-12	1,15E-12						
2K=36	50	1,11E-08	1,10E-08	50	7,61E-12	7,75E-12	39	8,66E-13	9,26E-13						
2K=38	48	1,35E-08	1,35E-08	50	7,97E-12	7,95E-12	39	3,24E-12	3,44E-12						
2K=40	48	6,07E-09	6,06E-09	50	8,03E-12	8,02E-12	39	3,24E-12	3,44E-12						
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2K=2	37	3,11E-15	3,55E-15	29	4,00E-15	4,88E-15	22	3,55E-15	4,44E-15	16	4,44E-15	5,33E-15	10	1,33E-15	2,66E-15
2K=4	37	3,11E-15	3,55E-15	24	3,11E-15	3,11E-15	17	1,33E-15	1,33E-15	14	1,78E-15	1,33E-15	9	1,33E-15	1,33E-15
2K=6	29	4,44E-15	4,88E-15	22	1,33E-15	1,33E-15	15	2,22E-15	1,33E-15	12	1,33E-15	2,22E-15	8	1,33E-15	1,78E-15
2K=8	26	4,00E-15	4,88E-15	22	1,33E-15	1,33E-15	13	4,69E-13	5,07E-13	12	1,33E-15	2,22E-15	8	1,33E-15	1,78E-15
2K=10	23	2,22E-15	2,66E-15	20	1,33E-15	1,33E-15	13	3,11E-15	3,11E-15	12	1,33E-15	2,22E-15			
2K=12	24	3,11E-15	3,55E-15	20	3,11E-15	4,00E-15	13	3,11E-15	3,11E-15	12	1,33E-15	2,22E-15			
2K=14	24	3,95E-14	4,31E-14	19	4,88E-15	4,88E-15	13	3,11E-15	3,11E-15						
2K=16	24	6,93E-14	7,59E-14	22	8,88E-16	1,78E-15									
2K=18	24	4,44E-15	4,88E-15	20	2,42E+15	4,78E+08									
2K=20	23	1,11E-14	1,29E-14	20	2,42E+15	4,78E+08									
2K=22	23	1,48E-13	1,62E-13												
2K=24	23	1,48E-13	1,62E-13												

Table A.5: Example 4 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2K=2	50	1,46E-03	1,58E-03	50	5,65E-06	6,10E-06	50	9,92E-09	1,07E-08	50	6,31E-12	6,82E-12	49	2,22E-15	2,22E-15
2K=4	50	2,90E-04	3,01E-04	50	3,50E-08	3,66E-08	50	2,31E-12	2,51E-12	50	4,96E-13	5,37E-13	45	5,91E-10	6,39E-10
2K=6	50	4,12E-06	4,22E-06	50	1,09E-09	1,14E-09	50	3,02E-12	3,27E-12	50	2,56E-13	2,78E-13	36	2,22E-15	1,78E-15
2K=8	50	3,65E-04	3,73E-04	50	8,69E-11	9,08E-11	50	2,61E-12	2,81E-12	45	4,44E-15	4,00E-15	29	2,66E-15	3,11E-15
2K=10	50	1,33E-06	1,37E-06	50	4,74E-11	5,11E-11	50	4,72E-13	5,09E-13	46	1,78E-15	1,78E-15	33	1,95E-14	2,13E-14
2K=12	50	3,32E-07	3,41E-07	50	3,51E-11	3,68E-11	50	6,48E-13	6,99E-13	31	1,86E-13	1,97E-13	25	1,82E-14	1,73E-14
2K=14	50	6,56E-08	6,74E-08	50	4,50E-12	4,49E-12	50	1,71E-13	1,85E-13	31	1,59E-11	1,70E-11	25	2,26E-14	2,35E-14
2K=16	50	9,40E-08	9,69E-08	50	7,98E-12	7,97E-12	48	1,33E-15	1,33E-15	31	2,57E-12	2,77E-12	25	4,49E-14	4,88E-14
2K=18	50	2,74E-07	2,81E-07	50	4,64E-11	4,86E-11	48	1,33E-15	1,33E-15	31	2,53E-12	2,73E-12	25	3,29E-14	3,55E-14
2K=20	50	2,31E-07	2,37E-07	50	3,01E-11	3,13E-11	50	2,66E-15	3,55E-15	31	4,65E-13	4,75E-13	22	2,82E-13	3,05E-13
2K=22	50	5,10E-07	5,23E-07	50	3,44E-11	3,55E-11	50	9,45E-13	1,02E-12	31	5,47E-13	5,57E-13	22	2,82E-13	3,05E-13
2K=24	50	1,24E-07	1,27E-07	50	6,81E-11	7,04E-11	50	8,97E-14	9,68E-14	31	3,90E-13	4,06E-13			
2K=26	50	1,57E-07	1,62E-07	50	5,58E-11	5,78E-11	50	7,77E-14	8,48E-14	31	3,72E-13	3,90E-13			
2K=28	50	1,57E-07	1,61E-07	50	6,10E-12	6,09E-12	50	1,36E-13	1,44E-13	31	3,72E-13	3,90E-13			
2K=30	50	1,57E-07	1,61E-07	50	7,44E-12	7,42E-12	50	5,51E-14	5,91E-14	31	3,72E-13	3,90E-13			
2K=32	50	1,61E-07	1,65E-07	50	1,89E-11	1,98E-11	50	5,51E-14	5,91E-14	31	3,72E-13	3,90E-13			
2K=34	50	1,46E-07	1,50E-07	50	3,22E-11	3,34E-11	50	2,75E-14	2,84E-14						
2K=36	50	1,46E-07	1,50E-07	50	1,63E-11	1,77E-11	46	8,29E-13	8,48E-13						
2K=38	50	3,81E-07	3,92E-07	50	2,42E-11	2,56E-11	46	7,49E-13	7,63E-13						
2K=40	50	7,11E-08	7,37E-08	50	1,24E-10	1,28E-10	46	5,84E-13	5,96E-13						
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2K=2	37	3,55E-15	4,44E-15	29	3,11E-15	3,11E-15	22	3,11E-15	4,00E-15	16	4,44E-15	5,33E-15	10	1,78E-15	3,11E-15
2K=4	25	3,11E-15	2,22E-15	26	1,33E-15	8,88E-16	18	4,44E-15	4,00E-15	14	3,11E-15	2,66E-15	9	1,33E-15	1,78E-15
2K=6	25	3,82E-14	4,13E-14	23	2,22E-15	1,78E-15	15	2,66E-15	2,22E-15	12	1,78E-15	2,66E-15	8	8,88E-16	1,33E-15
2K=8	25	8,44E-15	8,44E-15	19	1,33E-15	1,78E-15	14	1,38E-14	1,47E-14	12	7,99E-15	9,33E-15	8	8,88E-16	1,33E-15
2K=10	24	8,88E-16	1,78E-15	19	1,33E-15	1,78E-15	14	2,04E-14	2,13E-14	12	4,00E-15	4,88E-15			
2K=12	23	1,64E-14	1,78E-14	17	1,33E-15	1,33E-15	13	2,22E-14	2,35E-14	12	4,00E-15	4,88E-15			
2K=14	22	1,60E-14	1,78E-14	16	2,44E-14	2,62E-14	13	2,22E-14	2,35E-14						
2K=16	22	5,46E-14	5,91E-14	16	2,44E-14	2,62E-14									
2K=18	22	2,89E-14	3,15E-14												
2K=20	22	2,18E-14	2,35E-14												
2K=22	22	2,18E-14	2,35E-14												

Table A.6: Example 4 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$	
2k=2	15	0,116986	0,116102	15	0,027238	0,026078	15	0,004616	0,004334	15	0,000582	0,00054	15	5,09E-05	4,68E-05
2k=4	15	0,051818	0,048374	15	0,013984	0,013126	15	0,004065	0,003811	15	0,000722	0,000676	15	8,89E-06	8,29E-06
2k=6	15	0,036122	0,034029	15	0,004399	0,004137	15	9,62E-06	9,12E-06	15	1,31E-06	1,22E-06	15	7,96E-08	7,50E-08
2k=8	15	0,03922	0,036953	15	0,001182	0,001113	15	3,10E-05	2,92E-05	15	1,99E-06	1,88E-06	15	7,63E-08	7,18E-08
2k=10	15	0,010952	0,01033	15	5,29E-05	4,99E-05	15	5,13E-07	4,98E-07	15	6,16E-07	5,81E-07	15	2,98E-08	2,80E-08
2k=12	15	0,000142	0,000132	15	5,90E-05	5,56E-05	15	5,09E-06	4,80E-06	15	6,75E-07	6,37E-07	15	8,79E-09	8,28E-09
2k=14	15	0,013005	0,012275	15	1,08E-05	1,02E-05	15	3,54E-06	3,34E-06	15	3,10E-07	2,93E-07	15	3,71E-10	3,50E-10
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$	
2k=4	15	1,35E-07	1,26E-07	15	1,00E-09	9,25E-10	15	1,98E-12	1,80E-12	14	4,84E-15	4,19E-15	9	3,92E-15	4,71E-15
2k=6	15	2,75E-09	2,57E-09	15	1,94E-11	1,80E-11	15	1,93E-14	1,77E-14	13	8,33E-16	6,44E-16	9	3,89E-16	3,89E-16
2k=8	15	3,67E-09	3,44E-09	15	2,85E-12	2,66E-12	15	1,79E-15	1,66E-15	12	4,51E-15	4,02E-15	9	3,33E-16	4,44E-16
2k=10	15	1,23E-10	1,16E-10	15	2,49E-14	2,36E-14	15	1,27E-15	1,19E-15	12	6,11E-16	6,11E-16	9	3,33E-16	4,44E-16
2k=12	15	1,65E-11	1,55E-11	15	6,50E-14	6,05E-14	13	1,22E-15	1,28E-15	12	6,11E-16	6,11E-16			
2k=14	15	9,40E-12	8,85E-12	15	6,31E-14	5,87E-14	13	1,22E-15	1,28E-15						

Table A.7: Example 20 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$	
2k=2	15	0,117535	0,117428	15	0,027595	0,026654	15	0,004721	0,004479	15	0,000601	0,000564	15	5,29E-05	4,93E-05
2k=4	15	0,051813	0,048805	15	0,013801	0,01307	15	0,003692	0,003494	15	0,001302	0,00123	15	1,08E-05	1,02E-05
2k=6	15	0,036418	0,034596	15	0,004765	0,004521	15	1,11E-05	1,06E-05	15	1,53E-06	1,44E-06	15	9,21E-08	8,75E-08
2k=8	15	0,037775	0,035887	15	0,001403	0,001332	15	3,61E-05	3,42E-05	15	2,31E-06	2,19E-06	15	8,84E-08	8,39E-08
2k=10	15	0,011984	0,011395	15	6,13E-05	5,83E-05	15	5,85E-07	5,70E-07	15	7,13E-07	6,78E-07	15	3,45E-08	3,27E-08
2k=12	15	0,002525	0,0024	15	6,85E-05	6,51E-05	15	5,88E-06	5,60E-06	15	7,80E-07	7,42E-07	15	1,02E-08	9,71E-09
2k=14	15	0,019784	0,018828	15	1,39E-05	1,32E-05	15	4,09E-06	3,89E-06	15	3,59E-07	3,41E-07	15	4,70E-10	4,47E-10
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$	
2k=2	15	2,75E-06	2,54E-06	15	6,15E-08	5,60E-08	15	2,85E-10	2,51E-10	15	2,42E-14	1,95E-14	12	7,77E-16	5,00E-16
2k=4	15	1,60E-07	1,50E-07	15	1,18E-09	1,10E-09	15	2,34E-12	2,16E-12	15	6,11E-16	5,13E-16	9	3,87E-15	4,65E-15
2k=6	15	3,19E-09	3,02E-09	15	2,26E-11	2,12E-11	15	2,29E-14	2,13E-14	13	7,77E-16	7,36E-16	9	4,44E-16	3,89E-16
2k=8	15	4,26E-09	4,03E-09	15	3,32E-12	3,13E-12	15	1,37E-15	1,31E-15	12	4,70E-15	4,33E-15	9	4,44E-16	3,89E-16
2k=10	15	1,43E-10	1,35E-10	15	2,11E-14	2,03E-14	14	1,96E-15	1,80E-15	12	6,11E-16	6,11E-16	9	4,44E-16	3,89E-16
2k=12	15	1,96E-11	1,86E-11	15	8,46E-14	7,94E-14	14	1,17E-15	1,22E-15	12	6,11E-16	6,11E-16			
2k=14	15	1,14E-11	1,08E-11	15	8,46E-14	7,94E-14	14	1,17E-15	1,22E-15						

Table A.8: Example 20 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2k=2	50	2,26E-03	2,39E-03	50	7,93E-06	8,05E-06	50	1,68E-08	1,75E-08	50	1,77E-11	1,88E-11	50	8,22E-15	8,44E-15
2k=4	50	2,69E-04	2,79E-04	50	1,10E-06	1,15E-06	50	1,43E-09	1,54E-09	50	1,01E-12	1,06E-12	50	1,04E-14	1,07E-14
2k=6	50	1,39E-05	1,41E-05	50	9,34E-08	9,31E-08	50	3,37E-10	3,62E-10	50	9,91E-13	1,03E-12	50	1,69E-14	1,75E-14
2k=8	50	1,54E-05	1,56E-05	50	8,67E-08	8,80E-08	50	2,87E-10	2,94E-10	50	1,93E-12	2,03E-12	50	1,69E-14	1,75E-14
2k=10	50	1,59E-05	1,61E-05	50	9,99E-08	1,01E-07	50	2,64E-10	2,67E-10	50	3,85E-13	3,93E-13	49	4,44E-15	4,66E-15
2k=12	50	1,49E-05	1,51E-05	50	9,52E-08	9,64E-08	50	3,12E-10	3,15E-10	50	4,31E-13	4,39E-13	50	4,36E-14	4,60E-14
2k=14	36	4,97E-05	5,03E-05	50	1,29E-07	1,31E-07	50	3,00E-10	3,04E-10	50	4,89E-13	5,14E-13	50	1,22E-13	1,30E-13
2k=16	36	1,01E-04	1,03E-04	50	5,03E-07	5,10E-07	50	1,13E-10	1,15E-10	50	8,73E-13	9,27E-13	45	5,46E-14	5,68E-14
2k=18	36	2,08E-04	2,11E-04	50	4,27E-08	4,32E-08	50	1,87E-10	1,89E-10	50	7,47E-13	7,49E-13	25	1,36E-10	1,35E-10
2k=20	36	2,68E-05	2,71E-05	50	3,09E-09	3,24E-09	50	7,32E-11	7,44E-11	50	1,66E-12	1,68E-12	25	1,48E-10	1,47E-10
2k=22	36	3,40E-05	3,44E-05	50	6,90E-09	6,93E-09	50	1,18E-10	1,20E-10	50	6,84E-13	6,86E-13	25	1,86E-08	1,85E-08
2k=24	36	1,12E-03	1,14E-03	50	2,05E-09	2,13E-09	50	5,72E-11	5,82E-11	50	4,75E-13	4,72E-13	25	1,47E-08	1,46E-08
2k=26	36	1,77E-04	1,79E-04	50	4,97E-09	4,96E-09	50	4,75E-10	4,80E-10	50	7,27E-13	7,35E-13	25	1,47E-08	1,46E-08
2k=28	36	8,77E-04	8,89E-04	50	4,98E-09	4,96E-09	50	5,33E-10	5,38E-10	35	4,74E-12	4,86E-12			
2k=30	36	1,50E-02	1,52E-02	50	6,18E-09	6,16E-09	50	4,49E-10	4,54E-10	35	3,15E-11	3,14E-11			
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2k=2	40	3,33E-15	3,11E-15	32	3,22E-15	4,22E-15	25	4,55E-15	4,44E-15	19	4,77E-15	4,44E-15	20	2,89E-15	2,22E-15
2k=4	38	1,35E-14	1,40E-14	32	4,11E-15	4,00E-15	26	2,22E-15	1,78E-15	18	3,66E-15	3,11E-15	15	2,22E-15	1,44E-15
2k=6	37	3,55E-15	3,11E-15	30	4,66E-15	5,00E-15	22	4,88E-15	5,11E-15	17	2,44E-15	2,00E-15	11	4,11E-15	4,44E-15
2k=8	36	2,49E-14	2,66E-14	30	4,55E-15	4,44E-15	23	3,33E-15	3,11E-15	17	2,44E-15	2,00E-15	12	1,55E-15	1,33E-15
2k=10	36	1,78E-13	1,85E-13	30	2,66E-15	3,11E-15	22	1,78E-15	1,89E-15	17	2,44E-15	2,00E-15	11	2,78E-15	2,00E-15
2k=12	30	2,67E-13	2,70E-13	31	2,22E-15	2,00E-15	23	1,67E-15	1,78E-15	14	1,13E-14	1,13E-14	11	2,78E-15	2,00E-15
2k=14	30	4,61E-14	4,86E-14	31	8,22E-15	8,99E-15	22	2,22E-15	2,22E-15	14	1,13E-14	1,13E-14			
2k=16	17	2,38E+08	2,36E+08	31	8,55E-15	9,55E-15	21	4,22E-15	4,44E-15						
2k=18	17	2,38E+08	2,36E+08	23	2,15E-12	2,18E-12	21	4,22E-15	4,44E-15						
2k=20				23	2,56E-13	2,60E-13	23	4,91E-12	5,00E-12						
2k=22				23	5,00E-14	5,04E-14	23	2,30E-11	2,34E-11						
2k=24				23	5,00E-14	5,04E-14	23	2,30E-11	2,34E-11						

Table A.9: Example 21 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)}$ random.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2k=2	50	2,18E-03	2,27E-03	50	7,97E-06	8,11E-06	50	1,69E-08	1,76E-08	50	1,78E-11	1,88E-11	50	8,44E-15	8,66E-15
2k=4	50	2,87E-04	2,96E-04	50	1,24E-06	1,30E-06	50	1,71E-09	1,84E-09	50	8,56E-13	8,62E-13	50	4,77E-15	4,44E-15
2k=6	50	1,92E-05	1,93E-05	50	9,26E-08	9,26E-08	50	4,78E-09	5,10E-09	50	9,08E-13	9,16E-13	50	1,11E-14	1,15E-14
2k=8	50	2,18E-05	2,18E-05	50	4,16E-08	4,30E-08	50	3,34E-10	3,55E-10	50	1,56E-12	1,65E-12	50	1,11E-14	1,15E-14
2k=10	50	2,26E-05	2,27E-05	50	1,02E-07	1,02E-07	50	5,28E-10	5,36E-10	50	1,95E-12	2,06E-12	50	6,13E-14	6,57E-14
2k=12	50	2,43E-05	2,43E-05	50	1,31E-07	1,31E-07	50	4,54E-10	4,56E-10	50	3,94E-12	4,22E-12	50	2,12E-14	2,26E-14
2k=14	50	2,50E-05	2,51E-05	50	3,25E-07	3,25E-07	50	4,79E-10	4,79E-10	50	7,04E-13	7,13E-13	50	9,66E-15	1,02E-14
2k=16	50	2,68E-05	2,68E-05	50	1,45E-07	1,45E-07	50	5,48E-10	5,49E-10	50	3,52E-13	3,63E-13	49	1,01E-14	1,08E-14
2k=18	50	2,75E-05	2,75E-05	50	2,73E-07	2,73E-07	50	6,68E-10	6,69E-10	50	3,56E-13	3,71E-13	47	1,87E-14	1,87E-14
2k=20	50	2,61E-05	2,62E-05	50	2,82E-07	2,82E-07	50	1,89E-09	1,89E-09	50	1,66E-12	1,66E-12	37	1,84E-12	1,84E-12
2k=22	50	3,17E-05	3,17E-05	50	5,12E-07	5,13E-07	50	1,11E-09	1,11E-09	50	1,66E-12	1,66E-12	37	2,39E-10	2,39E-10
2k=24	50	3,19E-05	3,19E-05	50	6,12E-07	6,12E-07	50	1,12E-08	1,12E-08	50	3,77E-12	3,78E-12	37	6,87E-12	6,86E-12
2k=26	50	1,99E-05	1,99E-05	50	3,64E-07	3,64E-07	50	3,31E-10	3,31E-10	44	7,09E-11	7,09E-11	37	3,09E-11	3,09E-11
2k=28	50	2,10E-05	2,10E-05	50	5,49E-07	5,49E-07	50	3,59E-09	3,60E-09	44	8,44E-10	8,45E-10	37	8,87E-12	8,88E-12
2k=30	50	2,25E-05	2,26E-05	50	2,04E-07	2,04E-07	50	8,69E-10	8,70E-10	44	7,40E-10	7,41E-10	37	4,01E-13	4,00E-13
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2k=2	40	3,89E-15	5,11E-15	32	4,00E-15	4,00E-15	25	4,33E-15	4,22E-15	19	4,77E-15	4,44E-15	20	2,89E-15	2,22E-15
2k=4	41	2,78E-15	2,66E-15	32	4,11E-15	4,00E-15	25	1,23E-07	1,30E-07	16	3,11E-15	2,89E-15	15	1,89E-15	1,33E-15
2k=6	41	2,78E-15	2,66E-15	31	1,87E-14	2,00E-14	22	4,22E-15	4,22E-15	16	3,11E-15	2,89E-15	11	1,78E-15	2,33E-15
2k=8	42	3,00E-15	2,89E-15	30	4,44E-15	4,44E-15	21	4,66E-14	4,75E-14	16	3,11E-15	2,89E-15	11	1,89E-15	2,00E-15
2k=10	39	2,00E-15	1,67E-15	26	4,51E-14	4,53E-14	21	4,66E-14	4,75E-14	18	2,22E-15	1,55E-15	11	1,89E-15	2,00E-15
2k=12	40	2,44E-15	2,00E-15	25	3,78E-13	3,78E-13	21	4,06E-13	4,06E-13	18	3,44E-15	2,89E-15	11	1,89E-15	2,00E-15
2k=14	39	1,89E-15	1,78E-15	25	3,78E-13	3,78E-13	21	7,96E-12	7,96E-12	18	2,89E-15	3,11E-15			
2k=16	39	4,00E-15	4,00E-15	25	7,66E-13	7,64E-13	21	1,50E-12	1,50E-12	19	2,22E-15	1,33E-15			
2k=18	37	9,81E-14	9,90E-14	25	9,47E-13	9,47E-13	21	1,66E-12	1,66E-12	20	3,55E-15	4,44E-15			
2k=20	37	9,81E-14	9,90E-14	25	4,17E-12	4,16E-12	21	3,88E-12	3,88E-12	20	3,55E-15	4,44E-15			
2k=22	29	1,02E-11	1,02E-11	25	3,75E-12	3,74E-12	21	3,88E-12	3,88E-12						
2k=24	29	1,02E-11	1,02E-11	25	1,40E-13	1,41E-13									
2k=26	29	7,95E-12	7,96E-12	25	1,40E-13	1,41E-13									
2k=28	29	1,26E-11	1,26E-11												
2k=30	29	1,26E-11	1,26E-11												

Table A.10: Example 21 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)}$ random.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2k=2	50	3,98E-04	3,50E-04	50	5,04E-10	4,51E-10	43	1,61E-15	1,55E-15	26	1,39E-15	1,33E-15	4	3,94E-15	3,89E-15
2k=4	50	2,91E-11	2,58E-11	50	1,22E-12	1,10E-12	29	3,83E-15	3,33E-15	6	3,22E-15	3,33E-15	4	3,94E-15	3,89E-15
2k=6	50	3,30E-12	2,94E-12	28	4,27E-15	3,44E-15	7	1,05E-15	9,99E-16	6	3,22E-15	3,33E-15			
2k=8	50	2,06E-13	1,89E-13	50	7,86E-14	7,33E-14	7	1,05E-15	9,99E-16						
2k=10	50	2,05E-13	1,88E-13	18	2,83E-15	2,50E-15									
2k=12	18	2,33E-15	2,33E-15	23	3,94E-15	3,55E-15									
2k=14	34	3,77E-15	3,77E-15	16	3,00E-15	2,66E-15									
2k=16	20	2,00E-15	2,00E-15	16	3,00E-15	2,66E-15									
2k=18	26	3,44E-15	3,44E-15												
2k=20	25	4,55E-15	4,72E-15												
2k=22	25	3,61E-15	3,89E-15												
2k=24	26	1,44E-15	1,33E-15												
2k=26	26	1,44E-15	1,33E-15												
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2k=2	25	1,17E-15	1,05E-15	29	2,66E-15	3,11E-15	22	3,61E-15	3,28E-15	17	1,03E-15	5,00E-16	4	6,66E-16	5,55E-16
2k=4	5	2,05E-15	1,89E-15	5	5,55E-16	3,33E-16	5	6,66E-16	4,44E-16	5	5,55E-16	4,44E-16	4	6,66E-16	5,55E-16
2k=6	5	2,05E-15	1,89E-15	5	5,55E-16	3,33E-16	5	6,66E-16	4,44E-16	5	5,55E-16	4,44E-16			
2k=8															

Table A.11: Example 22 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2k=2	50	3,53E-04	3,04E-04	50	4,37E-10	3,84E-10	41	1,22E-15	1,11E-15	26	1,72E-15	1,55E-15	4	1,28E-15	1,22E-15
2k=4	50	3,27E-11	2,85E-11	50	9,51E-13	8,44E-13	17	2,28E-15	2,89E-15	10	3,72E-15	3,44E-15	4	1,28E-15	1,22E-15
2k=6	50	3,81E-12	3,37E-12	30	3,55E-15	4,00E-15	19	4,33E-15	4,44E-15	7	4,61E-15	4,33E-15			
2k=8	32	2,33E-15	2,11E-15	46	3,50E-15	3,05E-15	11	4,77E-15	4,66E-15	7	4,61E-15	4,33E-15			
2k=10	50	8,14E-13	7,33E-13	12	7,77E-16	5,55E-16	12	9,44E-16	8,33E-16						
2k=12	50	5,27E-13	4,75E-13	12	7,77E-16	5,55E-16	12	9,44E-16	8,33E-16						
2k=14	29	1,45E-13	1,30E-13												
2k=16	29	1,44E-13	1,29E-13												
2k=18	22	4,55E-15	3,77E-15												
2k=20	24	1,55E-15	1,39E-15												
2k=22	26	2,33E-15	2,11E-15												
2k=24	27	3,89E-15	3,89E-15												
2k=26	28	3,44E-15	3,44E-15												
2k=28	28	3,44E-15	3,44E-15												
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2k=2	25	2,22E-15	2,39E-15	29	2,72E-15	3,16E-15	22	3,66E-15	3,28E-15	17	8,60E-16	5,00E-16	4	6,66E-16	5,55E-16
2k=4	5	2,28E-15	1,89E-15	5	6,66E-16	4,44E-16	5	7,22E-16	4,44E-16	5	6,66E-16	4,44E-16	4	6,66E-16	5,55E-16
2k=6	5	2,28E-15	1,89E-15	5	6,66E-16	4,44E-16	5	7,22E-16	4,44E-16	5	6,66E-16	4,44E-16			
2k=8															

Table A.12: Example 22 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2k=2	30	5,20E-03	6,33E-03	30	2,36E-04	1,14E-03	30	7,93E-06	9,00E-04	30	2,29E-07	8,92E-04	30	2,37E-09	8,92E-04
2k=4	30	3,00E-10	8,92E-04	30	1,32E-11	8,92E-04	30	2,81E-13	8,92E-04	30	1,47E-13	8,92E-04	30	4,04E-14	8,92E-04
2k=6	30	6,28E-11	8,92E-04	30	5,52E-12	8,92E-04	30	2,81E-13	8,92E-04	30	1,76E-13	8,92E-04	8	4,44E-16	8,92E-04
2k=8	30	5,21E-12	8,92E-04	30	1,34E-13	8,92E-04	30	1,59E-14	8,92E-04	9	3,33E-15	8,92E-04	8	4,44E-16	8,92E-04
2k=10	30	6,64E-12	8,92E-04	30	5,26E-13	8,92E-04	17	2,33E-15	8,92E-04	9	3,33E-15	8,92E-04			
2k=12	30	4,25E-13	8,92E-04	16	2,11E-15	8,92E-04	13	1,89E-15	8,92E-04						
2k=14	30	6,27E-13	8,92E-04	17	9,99E-16	8,92E-04	13	1,89E-15	8,92E-04						
2k=16	30	7,81E-13	8,92E-04	19	4,11E-15	8,92E-04									
2k=18	30	6,13E-13	8,92E-04	25	2,00E-15	8,92E-04									
2k=20	30	4,80E-13	8,92E-04	26	4,55E-15	8,92E-04									
2k=22	30	4,83E-13	8,92E-04	27	1,05E-14	8,92E-04									
2k=24	30	1,63E-13	8,92E-04	27	1,49E-14	8,92E-04									
2k=26	30	1,91E-13	8,92E-04	27	1,22E-15	8,92E-04									
2k=28	30	6,58E-14	8,92E-04	27	1,22E-15	8,92E-04									
2k=30	30	6,58E-14	8,92E-04												
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2k=2	30	9,32E-13	8,92E-04	28	2,11E-15	8,92E-04	21	2,22E-15	8,92E-04	15	3,55E-15	8,92E-04	4	2,22E-16	8,92E-04
2k=4	29	2,22E-15	8,92E-04	8	3,33E-15	8,92E-04	5	1,78E-15	8,92E-04	5	8,88E-16	8,92E-04	4	2,22E-16	8,92E-04
2k=6	8	3,33E-15	8,92E-04	7	7,77E-16	8,92E-04	5	1,78E-15	8,92E-04	5	8,88E-16	8,92E-04			
2k=8	8	3,33E-15	8,92E-04	7	7,77E-16	8,92E-04									
2k=10															

Table A.13: Example 23 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$		
2k=2	30	5,13E-03	6,19E-03	30	2,18E-04	1,12E-03	30	6,40E-06	8,99E-04	30	1,23E-07	8,92E-04	30	1,92E-09	8,92E-04
2k=4	30	6,10E-10	8,92E-04	30	5,58E-12	8,92E-04	30	2,86E-12	8,92E-04	30	7,81E-13	8,92E-04	18	1,44E-15	8,92E-04
2k=6	30	3,70E-10	8,92E-04	30	1,05E-12	8,92E-04	30	2,96E-13	8,92E-04	30	5,41E-14	8,92E-04	15	1,44E-15	8,92E-04
2k=8	30	7,53E-12	8,92E-04	30	1,08E-12	8,92E-04	30	5,32E-14	8,92E-04	23	5,00E-15	8,92E-04	10	3,22E-15	8,92E-04
2k=10	30	1,56E-11	8,92E-04	29	1,11E-15	8,92E-04	14	2,22E-15	8,92E-04	14	2,89E-15	8,92E-04	10	3,22E-15	8,92E-04
2k=12	30	1,91E-12	8,92E-04	21	2,22E-15	8,92E-04	26	4,55E-15	8,92E-04	13	1,22E-15	8,92E-04			
2k=14	30	2,65E-12	8,92E-04	30	3,15E-12	8,92E-04	22	9,99E-16	8,92E-04	13	1,22E-15	8,92E-04			
2k=16	30	2,60E-12	8,92E-04	17	2,78E-15	8,92E-04	22	1,55E-15	8,92E-04						
2k=18	30	9,25E-12	8,92E-04	17	2,78E-15	8,92E-04	20	4,00E-15	8,92E-04						
2k=20	30	3,25E-13	8,92E-04				20	4,00E-15	8,92E-04						
2k=22	30	6,90E-13	8,92E-04												
2k=24	30	6,90E-13	8,92E-04												
2k=26	30	2,80E-13	8,92E-04												
2k=28	30	3,33E-13	8,92E-04												
2k=30	30	3,33E-13	8,92E-04												
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$		
2k=2	30	2,88E-12	8,92E-04	28	2,66E-15	8,92E-04	21	4,22E-15	8,92E-04	15	4,88E-15	8,92E-04	4	2,22E-16	8,92E-04
2k=4	7	1,78E-15	8,92E-04	5	4,00E-15	8,92E-04	5	2,44E-15	8,92E-04	5	5,55E-16	8,92E-04	4	2,22E-16	8,92E-04
2k=6	7	1,67E-15	8,92E-04	5	4,00E-15	8,92E-04	5	2,44E-15	8,92E-04	5	5,55E-16	8,92E-04			
2k=8	7	1,67E-15	8,92E-04												
2k=10															

Table A.14: Example 23 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$	
2k=2	30	6,94E-02	8,53E-02	30	2,80E-03	3,60E-03	30	2,91E-05	2,86E-05	30	4,59E-07	4,65E-07	30	2,74E-09	9,36E-09
2k=4	30	7,55E-03	4,29E-03	30	4,85E-04	4,73E-04	30	1,44E-05	1,44E-05	30	1,33E-07	1,39E-07	30	1,07E-09	7,69E-09
2k=6	30	5,00E-03	4,84E-03	30	1,33E-04	1,33E-04	30	3,53E-06	3,54E-06	30	9,85E-08	1,05E-07	30	1,75E-09	8,38E-09
2k=8	30	4,16E-03	4,22E-03	30	6,74E-05	6,88E-05	30	3,09E-06	3,10E-06	30	1,53E-07	1,59E-07	30	2,59E-08	2,31E-08
2k=10	30	3,78E-03	3,67E-03	30	7,45E-05	7,51E-05	30	4,03E-06	4,03E-06	30	1,01E-07	1,08E-07	30	1,77E-09	8,39E-09
2k=12	30	4,06E-03	3,97E-03	30	8,59E-05	8,64E-05	30	2,78E-06	2,79E-06	30	1,11E-07	1,18E-07	30	3,24E-09	9,86E-09
2k=14	30	3,55E-03	3,46E-03	30	8,85E-05	8,91E-05	30	3,03E-06	3,04E-06	30	8,03E-08	8,70E-08	30	1,80E-09	8,42E-09
2k=16	30	1,02E-03	7,27E-04	30	5,17E-05	5,17E-05	30	2,82E-06	2,82E-06	30	8,77E-08	9,43E-08	30	1,80E-09	8,42E-09
2k=18	30	2,95E-03	2,82E-03	30	6,89E-05	6,92E-05	30	2,70E-06	2,71E-06	19	8,90E-06	8,91E-06	30	1,37E-09	7,99E-09
2k=20	30	2,64E-03	2,50E-03	30	7,07E-05	7,09E-05	24	2,78E-04	2,76E-04	19	8,90E-06	8,91E-06	30	3,76E-09	1,04E-08
2k=22	30	2,78E-03	3,07E-03	30	1,28E-04	1,29E-04	24	2,01E-04	2,01E-04				28	8,58E-08	8,31E-08
2k=24	30	2,82E-03	2,68E-03	30	1,10E-04	1,11E-04	24	2,01E-04	2,01E-04				28	1,17E-07	1,14E-07
2k=26	30	3,19E-03	3,08E-03	30	2,66E-04	2,70E-04							28	3,41E-08	3,13E-08
2k=28	30	9,56E-04	1,20E-03	30	7,12E-04	7,26E-04							28	3,41E-08	3,13E-08
2k=30	30	9,56E-04	1,20E-03	30	7,12E-04	7,26E-04									
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$	
2k=2	30	5,21E-12	6,63E-09	30	3,33E-15	6,62E-09	25	2,55E-15	6,62E-09	18	2,22E-15	6,62E-09	17	1,11E-15	6,62E-09
2k=4	24	6,73E-10	7,29E-09	27	1,41E-13	6,62E-09	22	4,22E-15	6,62E-09	18	1,68E-10	6,45E-09	14	6,66E-16	6,62E-09
2k=6	24	1,40E-09	8,02E-09	27	4,79E-13	6,62E-09	23	1,44E-15	6,62E-09	18	1,02E-14	6,62E-09	13	1,22E-15	6,62E-09
2k=8	24	1,12E-09	7,75E-09	23	2,26E-11	6,64E-09	21	4,74E-13	6,62E-09	15	3,25E-12	6,63E-09	13	1,22E-15	6,62E-09
2k=10	24	7,94E-09	1,46E-08	23	4,97E-11	6,67E-09	14	9,19E-10	7,54E-09	15	1,56E-13	6,62E-09	13	1,22E-15	6,62E-09
2k=12	24	1,11E-09	7,73E-09	20	3,32E-10	6,95E-09	13	1,02E-08	1,68E-08	15	1,56E-13	6,62E-09	13	8,88E-15	6,62E-09
2k=14	24	2,63E-09	9,25E-09	20	7,90E-11	6,54E-09	13	1,02E-08	1,68E-08	15	9,31E-13	6,62E-09	13	8,88E-15	6,62E-09
2k=16	17	1,41E-07	1,47E-07	17	Inf	3,38E+12				15	9,31E-13	6,62E-09			
2k=18	17	1,41E-07	1,47E-07	17	Inf	3,38E+12									
2k=20															

Table A.15: Example 24 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\alpha = 0,1$			$\alpha = 0,2$			$\alpha = 0,3$			$\alpha = 0,4$			$\alpha = 0,5$	
2k=2	30	5,46E-02	5,79E-02	30	1,80E-03	1,84E-03	30	3,51E-05	3,52E-05	30	4,47E-07	4,53E-07	30	2,72E-09	9,34E-09
2k=4	30	1,72E-02	1,74E-02	30	5,85E-04	5,86E-04	30	1,48E-05	1,48E-05	30	1,34E-07	1,40E-07	30	9,05E-10	7,53E-09
2k=6	30	5,92E-03	5,93E-03	30	1,26E-04	1,26E-04	30	3,45E-06	3,46E-06	30	9,87E-08	1,05E-07	30	1,94E-09	8,57E-09
2k=8	30	3,70E-03	3,69E-03	30	5,14E-05	5,12E-05	30	3,08E-06	3,09E-06	30	1,43E-07	1,50E-07	30	4,62E-09	1,12E-08
2k=10	30	4,39E-03	4,40E-03	30	8,16E-05	8,16E-05	30	6,34E-06	6,35E-06	30	3,18E-08	3,85E-08	30	1,85E-09	8,47E-09
2k=12	30	4,83E-03	4,84E-03	30	8,20E-05	8,19E-05	30	2,25E-06	2,25E-06	30	4,71E-08	4,44E-08	30	1,37E-09	7,99E-09
2k=14	30	4,15E-03	4,16E-03	30	8,24E-05	8,23E-05	30	3,41E-06	3,42E-06	30	1,07E-07	1,13E-07	30	7,72E-10	5,85E-09
2k=16	30	3,13E-03	3,16E-03	30	6,63E-05	6,63E-05	30	2,01E-06	2,01E-06	30	1,08E-07	1,15E-07	30	2,72E-09	9,35E-09
2k=18	30	4,00E-03	4,01E-03	30	6,94E-05	6,94E-05	30	2,70E-06	2,71E-06	19	9,26E-06	9,27E-06	19	6,66E-07	6,73E-07
2k=20	30	4,11E-03	4,12E-03	30	6,80E-05	6,79E-05	24	2,00E-04	2,00E-04	19	9,26E-06	9,27E-06	19	6,66E-07	6,73E-07
2k=22	30	2,38E-03	2,40E-03	30	1,37E-04	1,37E-04	24	8,23E-05	8,24E-05						
2k=24	30	3,68E-03	3,69E-03	30	6,22E-06	6,37E-06	24	8,23E-05	8,24E-05						
2k=26	30	3,65E-03	3,66E-03	30	4,26E-05	4,27E-05									
2k=28	30	8,68E-04	7,96E-04	30	5,39E-05	5,39E-05									
2k=30	30	8,68E-04	7,96E-04	30	5,39E-05	5,39E-05									
	$\alpha = 0,6$			$\alpha = 0,7$			$\alpha = 0,8$			$\alpha = 0,9$			$\alpha = 1$	
2k=2	30	5,26E-12	6,63E-09	30	1,31E-13	6,62E-09	23	1,78E-15	6,62E-09	18	2,66E-15	6,62E-09	17	9,99E-16	6,62E-09
2k=4	30	8,67E-12	6,63E-09	23	1,96E-11	6,64E-09	15	5,99E-10	7,22E-09	17	2,55E-15	6,62E-09	14	1,33E-15	6,62E-09
2k=6	28	1,23E-10	6,75E-09	23	1,96E-11	6,64E-09	15	6,85E-10	7,31E-09	18	5,55E-16	6,62E-09	13	1,44E-15	6,62E-09
2k=8	28	9,64E-11	6,53E-09	19	1,07E-09	7,69E-09	15	3,60E-10	6,98E-09	17	1,92E-13	6,62E-09	13	1,44E-15	6,62E-09
2k=10	28	9,64E-11	6,53E-09	19	1,07E-09	7,69E-09	15	7,49E-10	7,37E-09	15	1,09E-12	6,62E-09	13	1,44E-15	6,62E-09
2k=12	28	9,64E-11	6,53E-09	18	2,28E-09	8,90E-09	15	3,69E-09	2,93E-09	15	1,09E-12	6,62E-09	14	1,89E-15	6,62E-09
2k=14	28	1,47E-10	6,77E-09	18	9,32E-10	7,55E-09	15	2,73E-09	3,89E-09	15	2,17E-12	6,62E-09	14	1,89E-15	6,62E-09
2k=16	17	1,42E-07	1,49E-07	18	9,32E-10	7,55E-09	15	2,73E-09	3,89E-09	15	2,17E-12	6,62E-09			
2k=18	17	1,42E-07	1,49E-07	18	9,32E-10	7,55E-09									
2k=20															

Table A.16: Example 24 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

Appendix B

Reference tables for adaptive step size strategy

In the following tables we report the number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector for every combination of $\bar{r}$ (vertical) init_start (horizontal). These is important to establish the optimal parameter for the fixed step size strategy that we discuss before.

When the value is not reported it means that the method has not produced a result or the previous one had already led to the best solution achievable for that column. We have computed the values reported in these tables by `comparison_for_adaptive_strategy.m` contained in the software pack that we wrote for thesis. For our experiments the tolerance parameter, Tol, is set to $5 \cdot 10^{-15}$. The other parameter of our adaptive step size strategy are set to: Tol_Sec $= 10^{-15}$ and Max_Iter_Sec $= 100$.

	$\bar{r} = 0,1$			$\bar{r} = 0,2$			$\bar{r} = 0,3$			$\bar{r} = 0,4$			$\bar{r} = 0,5$		
init_start = 1	12	1,44E-06	1,55E-06	17	3,80E-15	2,89E-15	20	4,17E-12	4,21E-12	26	1,19E-12	9,00E-13	33	1,40E-11	1,11E-11
init_start = 2	12	1,44E-06	1,55E-06	17	3,80E-15	2,89E-15	20	2,78E-16	2,78E-16	26	7,88E-15	7,33E-15	33	2,24E-11	1,84E-11
init_start = 3	12	1,44E-06	1,55E-06	17	1,67E-15	1,28E-15	19	4,16E-16	4,16E-16	26	7,88E-15	7,33E-15	33	5,70E-10	4,56E-10
init_start = 4	12	1,44E-06	1,55E-06	17	1,67E-15	1,28E-15	18	5,27E-16	5,27E-16	26	7,63E-15	6,49E-15	33	7,02E-12	5,61E-12
init_start = 5	12	1,44E-06	1,55E-06	18	1,19E-15	1,11E-15	17	1,72E-15	1,83E-15	25	1,86E-15	1,61E-15	33	7,50E-14	5,60E-14
init_start = 6	12	1,44E-06	1,55E-06	18	1,94E-15	1,39E-15	17	1,72E-15	1,83E-15	24	1,67E-16	1,94E-16	33	7,50E-14	5,60E-14
init_start = 7	12	1,44E-06	1,55E-06	17	1,67E-15	1,28E-15	17	9,16E-16	8,33E-16	23	1,11E-15	9,99E-16	33	8,77E-15	7,24E-15
init_start = 8	12	1,44E-06	1,55E-06	17	1,67E-15	1,28E-15	17	9,16E-16	8,33E-16	23	1,11E-15	9,99E-16	31	2,26E-13	1,78E-13
init_start = 9	15	1,44E-06	1,55E-06	19	1,94E-16	2,22E-16	17	5,55E-16	5,27E-16	23	2,78E-16	2,50E-16	31	4,41E-15	3,28E-15
init_start = 10	15	1,44E-06	1,55E-06	17	3,69E-15	2,78E-15	17	5,55E-16	5,27E-16	22	1,30E-15	1,28E-15	30	3,55E-15	3,39E-15
init_start = 11	15	1,44E-06	1,55E-06	17	2,64E-15	2,36E-15	17	9,16E-16	8,33E-16	21	3,30E-15	3,05E-15	31	3,03E-15	2,36E-15
init_start = 12	15	1,44E-06	1,55E-06	17	2,64E-15	2,36E-15	17	9,16E-16	8,33E-16	21	3,30E-15	3,05E-15	29	1,51E-12	1,18E-12
init_start = 13	15	1,44E-06	1,55E-06	18	1,30E-15	1,17E-15	18	5,27E-16	5,27E-16	21	1,64E-15	1,39E-15	33	1,82E-14	1,66E-14
init_start = 14	18	1,44E-06	1,55E-06	19	1,94E-16	2,22E-16	18	8,33E-17	8,33E-17	21	1,64E-15	1,39E-15	31	2,50E-15	1,78E-15
init_start = 15	18	1,44E-06	1,55E-06	20	5,55E-17	5,55E-17	18	8,33E-17	8,33E-17	21	4,80E-15	4,50E-15	35	1,08E-15	1,03E-15
	$\bar{r} = 0,6$			$\bar{r} = 0,7$			$\bar{r} = 0,8$			$\bar{r} = 0,9$			$\bar{r} = 1$		
init_start = 1	40	2,36E-09	1,82E-09	51	1,34E-09	1,09E-09	51	1,41E-06	1,45E-06	51	2,97E-06	2,73E-06	3	3.90E-01	4,00E-01
init_start = 2	40	1,58E-10	1,71E-10	52	1,01E-09	8,14E-10	52	1,21E-06	1,25E-06	52	2,96E-06	2,79E-06	4	3.90E-01	4,00E-01
init_start = 3	43	8,43E-11	6,66E-11	53	1,00E-09	8,47E-10	53	1,67E-06	1,75E-06	53	2,89E-06	2,68E-06	5	3.90E-01	4,00E-01
init_start = 4	41	1,79E-10	1,44E-10	54	6,59E-10	5,30E-10	54	1,02E-06	1,05E-06	54	2,61E-06	2,35E-06	6	3.90E-01	4,00E-01
init_start = 5	45	3,13E-11	2,51E-11	55	6,94E-10	5,61E-10	55	7,13E-07	7,31E-07	55	2,47E-06	2,25E-06	7	3.90E-01	4,00E-01
init_start = 6	45	6,56E-11	5,31E-11	56	5,85E-10	4,75E-10	56	5,05E-07	5,17E-07	56	2,46E-06	2,24E-06	8	3.90E-01	4,00E-01
init_start = 7	42	4,24E-11	2,43E-09	55	6,68E-10	5,42E-10	57	7,00E-07	7,29E-07	57	2,38E-06	2,17E-06	9	3.90E-01	4,00E-01
init_start = 8	45	5,17E-11	4,18E-11	52	1,56E-09	1,29E-09	58	6,29E-07	6,55E-07	58	1,94E-06	1,81E-06	10	3.90E-01	4,00E-01
init_start = 9	45	3,99E-11	3,18E-11	50	2,04E-09	1,73E-09	59	4,90E-07	5,09E-07	59	1,81E-06	1,71E-06	11	3.90E-01	4,00E-01
init_start = 10	45	1,52E-11	1,21E-11	54	6,24E-10	5,12E-10	60	3,89E-07	4,06E-07	60	2,15E-06	1,99E-06	12	3.90E-01	4,00E-01
init_start = 11	45	4,34E-13	3,34E-13	61	9,38E-11	7,64E-11	61	8,70E-08	8,86E-08	61	2,26E-06	2,08E-06	13	3.90E-01	4,00E-01
init_start = 12	45	9,89E-14	7,46E-14	62	7,20E-10	5,84E-10	62	3,14E-07	3,28E-07	62	2,14E-06	1,98E-06	14	3.90E-01	4,00E-01
init_start = 13	45	6,06E-14	4,99E-14	61	2,66E-10	2,16E-10	63	1,79E-06	1,90E-06	63	1,97E-06	1,83E-06	15	3.90E-01	4,00E-01
init_start = 14	45	5,28E-13	4,15E-13	55	1,80E-09	1,46E-09	63	1,79E-06	1,90E-06	64	1,49E-06	1,39E-06	16	3.90E-01	4,00E-01
init_start = 15	45	7,37E-14	5,47E-14	53	8,60E-10	6,72E-10	63	1,31E-07	1,33E-07	65	2,39E-06	2,22E-06	17	3.90E-01	4,00E-01

Table B.1: Example 2 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$.

	$\bar{r} = 0,1$			$\bar{r} = 0,2$			$\bar{r} = 0,3$			$\bar{r} = 0,4$			$\bar{r} = 0,5$		
init_start = 1	13	2,40E-07	2,46E-07	19	6,94E-16	6,94E-16	20	2,11E-10	2,11E-10	32	3,93E-12	4,06E-12	40	1,95E-12	2,10E-12
init_start = 2	13	2,40E-07	2,46E-07	18	5,83E-16	5,83E-16	20	8,77E-11	8,65E-11	32	4,62E-12	4,57E-12	40	2,59E-12	2,76E-12
init_start = 3	13	2,40E-07	2,46E-07	17	1,11E-15	1,17E-15	20	8,77E-11	8,65E-11	32	1,42E-12	1,41E-12	40	8,75E-11	6,61E-11
init_start = 4	13	2,40E-07	2,46E-07	16	2,78E-15	2,97E-15	20	1,12E-10	1,12E-10	32	2,13E-13	2,14E-13	40	2,77E-13	2,91E-13
init_start = 5	13	2,40E-07	2,46E-07	16	2,78E-15	2,97E-15	20	1,12E-10	1,12E-10	32	8,34E-14	8,39E-14	40	2,77E-13	2,91E-13
init_start = 6	13	2,40E-07	2,46E-07	17	1,42E-15	1,47E-15	20	2,69E-15	2,69E-15	32	1,11E-13	1,12E-13	40	1,50E-14	1,59E-14
init_start = 7	13	2,40E-07	2,46E-07	17	1,42E-15	1,47E-15	19	2,75E-15	2,78E-15	32	2,41E-14	2,43E-14	40	1,50E-14	1,59E-14
init_start = 8	13	2,40E-07	2,46E-07	16	1,42E-15	1,47E-15	19	2,75E-15	2,78E-15	32	1,78E-15	1,75E-15	40	1,17E-14	1,06E-14
init_start = 9	13	2,40E-07	2,46E-07	16	4,22E-15	4,58E-15	19	1,17E-15	1,22E-15	27	4,27E-15	4,14E-15	40	1,17E-14	1,06E-14
init_start = 10	13	2,40E-07	2,46E-07	15	2,75E-15	2,97E-15	18	2,80E-15	2,69E-15	26	1,20E-14	1,14E-14	40	9,63E-15	9,66E-15
init_start = 11	13	2,40E-07	2,46E-07	15	2,75E-15	2,97E-15	18	2,80E-15	2,69E-15	24	1,11E-12	7,86E-13	40	9,63E-15	9,66E-15
init_start = 12	14	2,40E-07	2,46E-07	17	1,11E-15	1,17E-15	18	2,80E-15	2,69E-15	32	1,47E-13	1,45E-13	40	9,49E-15	9,46E-15
init_start = 13	15	2,40E-07	2,46E-07	18	6,11E-16	6,11E-16	18	2,80E-15	2,69E-15	25	1,47E-15	1,44E-15	37	3,80E-15	2,72E-15
init_start = 14	16	2,40E-07	2,46E-07	19	6,38E-16	6,38E-16	19	1,14E-15	1,22E-15	24	3,52E-15	3,52E-15	37	3,80E-15	2,72E-15
init_start = 15	18	2,40E-07	2,46E-07	20	5,55E-17	1,11E-16	19	8,33E-16	8,60E-16	24	3,52E-15	3,52E-15	37	1,11E-15	1,08E-15
	$\bar{r} = 0,6$			$\bar{r} = 0,7$			$\bar{r} = 0,8$			$\bar{r} = 0,9$			$\bar{r} = 1$		
init_start = 1	36	9,49E-07	9,73E-07	25	7,30E-07	7,49E-07	43	6,75E-07	7,12E-07	41	3,16E-06	3,30E-06	3	3.90E-01	4,00E-01
init_start = 2	36	9,58E-07	9,82E-07	25	7,30E-07	7,49E-07	43	5,77E-07	6,03E-07	42	3,16E-06	3,30E-06	4	3.90E-01	4,00E-01
init_start = 3	37	7,63E-07	7,82E-07	25	7,35E-07	7,54E-07	43	3,42E-07	2,91E-07	47	1,94E-04	2,02E-04	5	3.90E-01	4,00E-01
init_start = 4	37	7,63E-07	7,82E-07	26	7,85E-07	8,04E-07	43	3,84E-07	3,96E-07	47	1,94E-04	2,02E-04	6	3.90E-01	4,00E-01
init_start = 5	35	1,07E-06	1,10E-06	45	2,50E-07	2,56E-07	43	4,04E-07	4,16E-07	47	1,48E-05	1,54E-05	7	3.90E-01	4,00E-01
init_start = 6	35	1,07E-06	1,10E-06	45	2,50E-07	2,56E-07	43	3,84E-07	3,97E-07	47	2,06E-06	2,16E-06	8	3.90E-01	4,00E-01
init_start = 7	35	1,46E-06	1,50E-06	45	2,01E-07	2,06E-07	43	3,91E-07	4,03E-07	47	6,63E-06	6,90E-06	9	3.90E-01	4,00E-01
init_start = 8	35	1,46E-06	1,50E-06	45	2,33E-07	2,39E-07	43	3,91E-07	4,04E-07	47	2,35E-06	2,46E-06	10	3.90E-01	4,00E-01
init_start = 9	35	6,65E-06	6,81E-06	45	4,39E-08	4,48E-08	43	3,86E-07	3,94E-07	47	1,36E-05	1,41E-05	11	3.90E-01	4,00E-01
init_start = 10	35	6,65E-06	6,81E-06	46	2,33E-08	2,37E-08	43	3,86E-07	3,94E-07	47	1,33E-05	1,39E-05	12	3.90E-01	4,00E-01
init_start = 11	35	7,46E-07	7,64E-07	45	4,67E-08	4,76E-08	43	8,86E-07	6,60E-07	47	7,52E-07	8,00E-07	13	3.90E-01	4,00E-01
init_start = 12	35	7,46E-07	7,64E-07	46	1,49E-08	1,55E-08	43	8,86E-07	6,60E-07	47	7,52E-07	8,00E-07	14	3.90E-01	4,00E-01
init_start = 13	35	1,72E-07	1,76E-07	52	1,45E-07	1,48E-07	43	9,45E-07	7,75E-07	63	1,32E-05	1,37E-05	15	3.90E-01	4,00E-01
init_start = 14	35	1,72E-07	1,76E-07	52	7,03E-08	7,18E-08	43	9,45E-07	7,75E-07	64	3,09E-07	3,23E-07	16	3.90E-01	4,00E-01
init_start = 15	35	1,21E-07	1,24E-07	52	7,03E-08	7,18E-08	43	7,34E-07	6,42E-07	47	1,63E-06	1,67E-06	17	3.90E-01	4,00E-01

Table B.2: Example 2 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv \frac{2}{3}\frac{4}{3\sqrt{2\pi}}$.

	$\bar{r} = 0,1$			$\bar{r} = 0,2$			$\bar{r} = 0,3$			$\bar{r} = 0,4$			$\bar{r} = 0,5$		
init_start = 1	8	2,86E-01	3,58E-01	3	3,67E-01	4,30E-01				5	9,38E-02	9,32E-02			
init_start = 2	8	2,86E-01	3,58E-01	3	3,67E-01	4,30E-01				5	9,38E-02	9,32E-02			
init_start = 3	10	2,86E-01	3,58E-01							5	2,16E-01	2,52E-01			
init_start = 4	10	2,86E-01	3,58E-01							5	2,16E-01	2,52E-01			
init_start = 5	10	2,86E-01	3,58E-01												
init_start = 6	10	2,86E-01	3,58E-01												
init_start = 7	10	2,86E-01	3,58E-01												
init_start = 8	10	2,86E-01	3,58E-01												
init_start = 9	13	2,86E-01	3,58E-01												
init_start = 10	13	2,86E-01	3,58E-01												
init_start = 11	13	2,86E-01	3,58E-01												
init_start = 12	16	2,86E-01	3,58E-01												
init_start = 13	16	2,86E-01	3,58E-01												
init_start = 14	16	2,86E-01	3,58E-01												
init_start = 15	17	2,86E-01	3,58E-01												
	$\bar{r} = 0,6$			$\bar{r} = 0,7$			$\bar{r} = 0,8$			$\bar{r} = 0,9$			$\bar{r} = 1$		
init_start = 1	36	4,77E-14	5,00E-14	6	3,51E-01	3,20E-01							3	1,36E+00	9,66E-01
init_start = 2	36	2,75E-14	2,62E-14	6	3,31E-01	3,05E-01							4	1,36E+00	9,66E-01
init_start = 3	30	2,75E-14	4,31E+07	6	3,31E-01	3,05E-01							5	1,36E+00	9,66E-01
init_start = 4	34	3,11E-15	2,64E-15	6	5,11E-01	5,17E-01							6	1,36E+00	9,66E-01
init_start = 5	27	3,57E-12	1,44E-12	6	5,11E-01	5,17E-01							7	1,36E+00	9,66E-01
init_start = 6	25	7,29E-11	3,08E-11										8	1,36E+00	9,66E-01
init_start = 7	31	3,77E-14	3,94E-14										9	1,36E+00	9,66E-01
init_start = 8	30	3,00E-15	1,98E-15										10	1,36E+00	9,66E-01
init_start = 9	30	3,00E-15	1,98E-15										11	1,36E+00	9,66E-01
init_start = 10	28	5,00E-15	3,44E-15										12	1,36E+00	9,66E-01
init_start = 11	28	5,00E-15	3,44E-15										13	1,36E+00	9,66E-01
init_start = 12	32	1,33E-15	3,61E-15										14	1,36E+00	9,66E-01
init_start = 13	32	2,66E-15	5,57E-15										15	1,36E+00	9,66E-01
init_start = 14	31	3,38E-15	2,11E-15										16	1,36E+00	9,66E-01
init_start = 15	31	3,11E-15	2,44E-15										17	1,36E+00	9,66E-01

Table B.3: Example 3 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r} = 0,1$			$\bar{r} = 0,2$			$\bar{r} = 0,3$			$\bar{r} = 0,4$			$\bar{r} = 0,5$		
init_start = 1	10	2,86E-01	3,58E-01	3	3,67E-01	4,30E-01	4	1,67E-01	1,53E-01	5	9,38E-02	9,32E-02	6	3,10E-01	3,43E-01
init_start = 2	10	2,86E-01	3,58E-01	3	3,67E-01	4,30E-01	4	2,65E-01	3,12E-01	5	9,38E-02	9,32E-02	6	1,02E-01	9,74E-02
init_start = 3	10	2,86E-01	3,58E-01				4	2,65E-01	3,12E-01	5	2,16E-01	2,52E-01	6	1,02E-01	9,74E-02
init_start = 4	10	2,86E-01	3,58E-01							5	2,16E-01	2,52E-01	6	2,30E-01	2,66E-01
init_start = 5	10	2,86E-01	3,58E-01										6	2,30E-01	2,66E-01
init_start = 6	10	2,86E-01	3,58E-01												
init_start = 7	10	2,86E-01	3,58E-01												
init_start = 8	10	2,86E-01	3,58E-01												
init_start = 9	12	2,86E-01	3,58E-01												
init_start = 10	12	2,86E-01	3,58E-01												
init_start = 11	15	2,86E-01	3,58E-01												
init_start = 12	15	2,86E-01	3,58E-01												
init_start = 13	15	2,86E-01	3,58E-01												
init_start = 14	16	2,86E-01	3,58E-01												
init_start = 15	17	2,86E-01	3,58E-01												
	$\bar{r} = 0,6$			$\bar{r} = 0,7$			$\bar{r} = 0,8$			$\bar{r} = 0,9$			$\bar{r} = 1$		
init_start = 1	30	6,22E-15	4,00E-15	43	2,92E-14	3,43E-14	51	2,59E-12	7,47E-13	38	2,54E+44	1,42E+02	3	1,36E+00	9,66E-01
init_start = 2	30	2,44E-14	9,33E-15	35	5,56E-13	3,42E-13	52	2,65E-12	2,60E-12	38	Inf	9,73E+02	4	1,36E+00	9,66E-01
init_start = 3	30	2,36E-13	1,29E-13	31	1,84E-13	7,14E-14	51	3,72E-12	2,79E-12	38	Inf	9,73E+02	5	1,36E+00	9,66E-01
init_start = 4	30	4,34E-13	3,48E-13	29	1,71E-13	7,92E-14	16	5,38E+62	1,47E+02	51	1,14E+19	4,63E+01	6	1,36E+00	9,66E-01
init_start = 5	27	3,35E-12	1,32E-12	34	4,59E-12	1,55E-12	16	5,38E+62	1,47E+02	55	2,31E-01	2,79E-01	7	1,36E+00	9,66E-01
init_start = 6	25	7,29E-11	3,08E-11	28	8,78E-11	3,36E-11	44	1,21E-11	4,36E-12	56	2,45E-01	2,95E-01	8	1,36E+00	9,66E-01
init_start = 7	31	3,11E-14	2,43E-14	43	5,65E-13	4,65E-13	42	7,44E-12	3,02E-12	57	1,91E+01	5,46E+00	9	1,36E+00	9,66E-01
init_start = 8	29	5,00E-15	7,37E-15	34	3,53E-13	1,36E-13	52	Inf	1,09E+09	58	1,99E-01	2,46E-01	10	1,36E+00	9,66E-01
init_start = 9	29	5,00E-15	7,37E-15	29	7,28E-12	3,15E-12	53	Inf	1,25E+13	59	1,34E-02	5,77E-03	11	1,36E+00	9,66E-01
init_start = 10	29	5,00E-15	7,37E-15	27	9,68E-11	4,27E-11	20	6,31E+10	2,73E+01	38	2,14E+05	1,46E+01	12	1,36E+00	9,66E-01
init_start = 11	32	2,22E-15	2,89E-15	32	1,11E-13	9,27E-14	20	6,31E+10	2,73E+01	38	2,14E+05	1,46E+01	13	1,36E+00	9,66E-01
init_start = 12	35	5,67E-13	6,26E-13	52	3,56E-13	4,43E-13	37	6,40E-10	3,16E-10	62	3,63E-04	4,05E-04	14	1,36E+00	9,66E-01
init_start = 13	33	1,63E-15	3,74E-15	41	2,33E-15	1,78E-15	47	6,31E+09	6,31E+09	63	4,56E-05	2,18E-05	15	1,36E+00	9,66E-01
init_start = 14	30	2,35E-15	2,66E-15	40	4,44E-15	4,22E-15	36	3,93E-10	1,88E-10	64	4,18E-06	1,93E-06	16	1,36E+00	9,66E-01
init_start = 15	30	2,35E-15	2,66E-15	35	1,04E-11	5,03E-12	58	7,13E-14	4,60E-14	65	1,71E-06	7,88E-07	17	1,36E+00	9,66E-01

Table B.4: Example 3 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r} = 0,1$			$\bar{r} = 0,2$			$\bar{r} = 0,3$			$\bar{r} = 0,4$			$\bar{r} = 0,5$		
init_start = 1	9	3,11E-15	3,55E-15				14	2,51E-11	2,51E-11				27	1,47E-13	1,46E-13
init_start = 2	9	3,11E-15	3,55E-15				14	2,13E-12	2,13E-12				27	1,48E-13	1,49E-13
init_start = 3	10	8,88E-16	1,33E-15				17	7,19E-14	7,28E-14				27	2,27E-12	2,27E-12
init_start = 4	10	2,22E-15	1,78E-15				18	2,98E-14	2,98E-14				27	8,02E-13	8,01E-13
init_start = 5	10	2,22E-15	1,78E-15				17	2,00E-14	2,13E-14				27	1,18E-10	1,18E-10
init_start = 6	11	1,78E-15	1,33E-15				18	1,42E-14	1,55E-14				27	2,34E-13	2,32E-13
init_start = 7	11	2,22E-15	3,11E-15				18	1,42E-14	1,55E-14				27	1,11E-13	1,10E-13
init_start = 8	11	2,22E-15	3,11E-15				18	1,42E-14	1,55E-14				27	8,83E-12	8,82E-12
init_start = 9	12	1,78E-15	8,88E-16				18	1,42E-14	1,55E-14				27	5,32E-12	5,31E-12
init_start = 10	13	1,33E-15	8,88E-16				18	1,12E-13	1,11E-13				27	5,74E-11	5,73E-11
init_start = 11	14	1,33E-15	8,88E-16				18	1,12E-13	1,11E-13				27	2,15E-13	2,14E-13
init_start = 12	15	8,88E-16	8,88E-16				18	8,88E-15	9,77E-15				27	2,15E-13	2,14E-13
init_start = 13	16	4,44E-16	1,33E-15				19	4,44E-15	4,00E-15				27	4,97E-13	4,97E-13
init_start = 14	16	4,00E-15	5,33E-15				19	4,44E-15	4,00E-15				27	4,97E-13	4,97E-13
init_start = 15	17	8,88E-16	1,33E-15				20	3,11E-15	4,00E-15				27	6,83E-13	6,81E-13
	$\bar{r} = 0,6$			$\bar{r} = 0,7$			$\bar{r} = 0,8$			$\bar{r} = 0,9$			$\bar{r} = 1$		
init_start = 1	6	1,25E-02	1,25E-02	10	1,79E-03	1,79E-03	3	1,73E+00	1,97E+00	4	1,61E-01	1,61E-01	3	2,67E+00	2,97E+00
init_start = 2	6	2,34E-02	2,34E-02	10	7,22E-03	7,20E-03	3	1,73E+00	1,97E+00	4	1,96E+00	2,22E+00	4	2,67E+00	2,97E+00
init_start = 3	6	2,34E-02	2,34E-02	10	7,22E-03	7,20E-03				4	1,96E+00	2,22E+00	5	2,67E+00	2,97E+00
init_start = 4	6	2,42E-01	2,82E-01	10	4,75E-03	4,74E-03							6	2,67E+00	2,97E+00
init_start = 5	6	2,42E-01	2,82E-01	10	4,75E-03	4,74E-03							7	2,67E+00	2,97E+00
init_start = 6				10	9,78E-03	9,76E-03							8	2,67E+00	2,97E+00
init_start = 7				10	9,78E-03	9,76E-03							9	2,67E+00	2,97E+00
init_start = 8				10	1,29E-01	1,51E-01							10	2,67E+00	2,97E+00
init_start = 9				10	1,29E-01	1,51E-01							11	2,67E+00	2,97E+00
init_start = 10													12	2,67E+00	2,97E+00
init_start = 11													13	2,67E+00	2,97E+00
init_start = 12													14	2,67E+00	2,97E+00
init_start = 13													15	2,67E+00	2,97E+00
init_start = 14													16	2,67E+00	2,97E+00
init_start = 15													17	2,67E+00	2,97E+00

Table B.5: Example 4 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$.

	$\bar{r} = 0,1$			$\bar{r} = 0,2$			$\bar{r} = 0,3$			$\bar{r} = 0,4$			$\bar{r} = 0,5$		
init_start = 1	9	4,00E-15	3,55E-15	12	7,11E-15	6,22E-15	19	2,31E-14	2,22E-14	23	9,02E-14	8,93E-14	24	2,39E-11	2,39E-11
init_start = 2	9	4,00E-15	3,55E-15	12	1,32E-11	1,31E-11	19	2,31E-14	2,22E-14	23	1,99E-13	1,98E-13	26	2,02E-13	2,00E-13
init_start = 3	10	1,78E-15	2,22E-15	12	7,11E-15	6,22E-15	18	2,39E-13	2,39E-13	23	3,57E-13	3,57E-13	26	1,90E-13	1,89E-13
init_start = 4	10	8,88E-16	8,88E-16	12	7,11E-15	6,22E-15	19	1,25E-13	1,26E-13	26	1,37E+25	5,38E+13	25	5,99E-12	5,98E-12
init_start = 5	10	8,88E-16	8,88E-16	13	2,22E-15	2,22E-15	19	2,66E-15	3,11E-15	21	3,11E-15	4,00E-15	26	3,43E-11	3,43E-11
init_start = 6	11	2,22E-15	1,78E-15	13	2,22E-15	2,22E-15	19	2,66E-15	3,11E-15	23	9,55E-14	9,46E-14	26	2,36E-13	2,35E-13
init_start = 7	11	2,22E-15	1,78E-15	13	2,22E-15	2,22E-15	19	3,55E-15	3,11E-15	23	5,37E-14	5,91E-14	26	2,39E-12	2,39E-12
init_start = 8	11	2,22E-15	1,78E-15	13	2,22E-15	2,22E-15	19	3,55E-15	3,11E-15	23	1,06E-13	1,05E-13	26	1,09E-12	1,08E-12
init_start = 9	12	1,78E-15	8,88E-16	14	1,33E-15	2,22E-15	21	3,11E-15	2,66E-15	23	9,55E-14	9,46E-14	32	3,11E-12	3,26E-12
init_start = 10	13	3,11E-15	2,66E-15	15	1,33E-15	8,88E-16	21	3,11E-15	2,66E-15	23	9,55E-14	9,46E-14	32	1,51E-10	1,50E-10
init_start = 11	14	1,33E-15	8,88E-16	16	2,22E-15	1,33E-15	23	2,22E-15	2,66E-15	23	1,40E-13	1,39E-13	32	2,93E-14	2,84E-14
init_start = 12	15	1,33E-15	8,88E-16	16	2,22E-15	1,33E-15	23	2,22E-15	3,11E-15	23	1,40E-13	1,39E-13	32	4,17E-12	4,18E-12
init_start = 13	16	1,78E-15	2,66E-15	16	2,22E-15	1,33E-15	22	2,66E-15	2,66E-15	23	5,37E-14	5,46E-14	32	4,17E-12	4,18E-12
init_start = 14	17	4,44E-16	1,33E-15	17	1,33E-15	2,22E-15	22	2,66E-15	2,66E-15	23	5,37E-14	5,46E-14	32	1,53E-13	1,52E-13
init_start = 15	17	8,88E-16	1,78E-15	18	8,88E-16	1,33E-15	22	2,66E-15	2,66E-15	23	3,13E-13	3,13E-13	32	1,53E-13	1,52E-13
	$\bar{r} = 0,6$			$\bar{r} = 0,7$			$\bar{r} = 0,8$			$\bar{r} = 0,9$			$\bar{r} = 1$		
init_start = 1	34	2,42E-09	2,41E-09	38	6,69E-11	6,67E-11	26	2,83E-06	2,83E-06	36	1,65E-04	1,65E-04	4	2,67E+00	2,97E+00
init_start = 2	29	7,57E-09	7,56E-09	38	3,80E-10	3,79E-10	52	1,18E-10	1,19E-10	36	1,49E-04	1,49E-04	4	2,67E+00	2,97E+00
init_start = 3	33	5,20E-11	5,19E-11	37	8,49E-11	8,48E-11	53	1,85E-10	1,85E-10	36	1,46E-04	1,46E-04	12	2,67E+00	2,97E+00
init_start = 4	36	7,75E-13	7,73E-13	38	2,01E-10	2,01E-10	54	1,72E-10	1,72E-10	36	1,49E-04	1,49E-04	12	2,67E+00	2,97E+00
init_start = 5	36	9,29E-13	9,26E-13	34	4,17E-08	4,16E-08	54	5,39E-10	5,38E-10	36	1,49E-04	1,49E-04	12	2,67E+00	2,97E+00
init_start = 6	36	4,22E-14	4,13E-14	37	6,16E-07	6,15E-07	36	1,97E-09	1,97E-09	36	5,08E-05	5,07E-05	12	2,67E+00	2,97E+00
init_start = 7	44	4,46E-12	4,45E-12	24	1,04E+19	3,41E+10	26	4,07E+19	6,73E+10	36	5,08E-05	5,07E-05	12	2,67E+00	2,97E+00
init_start = 8	44	1,37E-12	1,36E-12	38	8,79E-11	8,78E-11	58	1,94E-10	1,94E-10	36	5,83E-05	5,82E-05	12	2,67E+00	2,97E+00
init_start = 9	45	2,96E-10	2,95E-10	38	5,34E-11	5,33E-11	42	1,72E+26	1,39E+14	36	5,83E-05	5,82E-05	12	2,67E+00	2,97E+00
init_start = 10	40	3,62E-12	3,62E-12	38	3,65E-11	3,64E-11	60	2,19E-09	2,19E-09	36	5,04E-05	5,04E-05	12	2,67E+00	2,97E+00
init_start = 11	35	1,30E-12	1,30E-12	38	4,44E-11	4,43E-11	39	1,79E+19	4,46E+10	36	5,04E-05	5,04E-05	17	2,67E+00	2,97E+00
init_start = 12	45	1,10E-11	1,11E-11	38	3,12E-11	3,13E-11	40	1,76E-08	1,75E-08	36	4,95E-05	4,94E-05	17	2,67E+00	2,97E+00
init_start = 13	45	2,72E-11	2,72E-11	38	1,17E-10	1,16E-10	43	1,59E-08	1,59E-08	36	4,95E-05	4,94E-05	17	2,67E+00	2,97E+00
init_start = 14	45	4,85E-12	4,84E-12	29	1,26E+19	3,74E+10	49	7,06E-07	7,04E-07	36	6,00E-06	6,00E-06	17	2,67E+00	2,97E+00
init_start = 15	45	8,39E-14	8,30E-14	38	4,26E-10	4,25E-10	65	2,79E-12	2,78E-12	37	6,78E-05	6,77E-05	17	2,67E+00	2,97E+00

Table B.6: Example 4 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r}=0,1$		$\bar{r}=0,2$		$\bar{r}=0,3$		$\bar{r}=0,4$		$\bar{r}=0,5$	
init_start =1			11	1,26E-15 1,21E-15	16	7,22E-16 6,31E-16	20	9,16E-16 8,47E-16		
init_start =2			11	1,26E-15 1,21E-15	15	3,12E-15 3,07E-15	24	5,00E-16 3,89E-16		
init_start =3			12	6,66E-16 6,66E-16	15	5,55E-16 5,55E-16	23	1,46E-15 1,38E-15		
init_start =4			12	6,66E-16 6,66E-16	15	5,55E-16 5,55E-16	21	3,04E-15 2,91E-15		
init_start =5			12	6,66E-16 6,66E-16	16	4,16E-15 4,01E-15	25	6,64E-16 6,30E-16		
init_start =6			13	5,55E-16 3,89E-16	15	1,80E-15 1,72E-15	21	1,62E-15 1,52E-15		
init_start =7			13	5,55E-16 3,89E-16	16	7,22E-16 6,31E-16	19	2,68E-15 2,54E-15		
init_start =8			13	5,55E-16 3,89E-16	16	1,34E-15 1,26E-15	19	2,68E-15 2,54E-15		
init_start =9			13	1,36E-15 1,32E-15	16	1,34E-15 1,26E-15	26	7,53E-15 7,20E-15		
init_start =10			13	1,36E-15 1,32E-15	17	1,65E-15 1,60E-15	23	2,43E-15 2,38E-15		
	$\bar{r}=0,6$		$\bar{r}=0,7$		$\bar{r}=0,8$		$\bar{r}=0,9$		$\bar{r}=1$	
init_start =1	7	8,01E-03 7,71E-03					4	2,71E-01 2,39E-01	3	1,20E+00 1,40E+00
init_start =2	7	8,01E-03 7,71E-03					4	8,83E-01 1,05E+00	4	1,20E+00 1,40E+00
init_start =3	7	2,05E-02 1,90E-02							5	1,20E+00 1,40E+00
init_start =4	7	2,05E-02 1,90E-02							6	1,20E+00 1,40E+00
init_start =5	7	8,07E-02 1,08E-01							7	1,20E+00 1,40E+00
init_start =6									8	1,20E+00 1,40E+00
init_start =7									9	1,20E+00 1,40E+00
init_start =8									10	1,20E+00 1,40E+00
init_start =9									11	1,20E+00 1,40E+00
init_start =10									12	1,20E+00 1,40E+00

Table B.7: Example 20 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r}=0,1$		$\bar{r}=0,2$		$\bar{r}=0,3$		$\bar{r}=0,4$		$\bar{r}=0,5$	
init_start =1	11	1,37E-14 1,37E-14	11	2,92E-15 2,92E-15	16	2,98E-15 2,85E-15	24	1,05E-14 1,06E-14	21	Inf 3,57E+11
init_start =2	12	1,21E-15 1,16E-15	11	2,92E-15 2,92E-15	17	3,75E-15 3,81E-15	24	1,80E-14 1,74E-14	22	5,37E-13 5,13E-13
init_start =3	11	4,65E-15 4,43E-15	12	2,83E-15 2,77E-15	13	1,69E-15 1,63E-15	24	1,80E-14 1,75E-14	25	2,71E-12 2,60E-12
init_start =4	11	4,65E-15 4,43E-15	12	1,77E-15 1,72E-15	13	1,69E-15 1,63E-15	24	1,92E-14 1,85E-14	31	6,66E-16 4,44E-16
init_start =5	12	4,55E-16 4,44E-16	12	1,77E-15 1,72E-15	15	3,27E-15 2,85E-15	24	2,02E-15 1,95E-15	29	2,58E-14 2,47E-14
init_start =6	12	5,00E-16 3,89E-16	13	1,11E-15 1,06E-15	16	4,84E-15 4,80E-15	24	3,28E-14 3,16E-14	31	3,07E-15 2,86E-15
init_start =7	12	5,00E-16 3,89E-16	13	7,22E-16 6,11E-16	16	4,84E-15 4,80E-15	24	9,44E-15 9,12E-15	33	1,69E-14 1,64E-14
init_start =8	13	5,00E-16 4,44E-16	13	7,22E-16 6,11E-16	17	2,05E-15 1,98E-15	24	7,72E-14 7,41E-14	32	5,00E-16 5,00E-16
init_start =9	14	5,00E-16 3,33E-16	13	1,51E-15 1,47E-15	17	2,05E-15 1,98E-15	23	4,41E-15 4,13E-15	33	1,48E-13 1,43E-13
init_start =10	14	2,17E-15 2,02E-15	13	1,51E-15 1,47E-15	17	2,05E-15 1,98E-15	23	4,41E-15 4,13E-15	30	3,78E-15 3,59E-15
	$\bar{r}=0,6$		$\bar{r}=0,7$		$\bar{r}=0,8$		$\bar{r}=0,9$		$\bar{r}=1$	
init_start =1	18	2,57E-09 2,45E-09	31	1,34E-12 1,19E-12	26	1,08E-06 1,03E-06	31	2,38E-03 2,27E-03	3	1,20E+00 1,40E+00
init_start =2	27	2,59E-09 2,46E-09	32	6,63E-10 6,33E-10	26	1,29E-06 1,23E-06	32	8,22E-04 7,84E-04	4	1,20E+00 1,40E+00
init_start =3	33	4,51E-14 4,27E-14	27	2,41E-09 2,43E-09	25	9,05E-07 8,64E-07	33	1,82E-02 1,73E-02	11	1,20E+00 1,40E+00
init_start =4	34	2,96E-13 2,82E-13	22	7,45E-08 7,11E-08	25	9,05E-07 8,64E-07	34	3,01E-03 2,87E-03	11	1,20E+00 1,40E+00
init_start =5	31	2,48E-11 2,37E-11	35	6,46E-10 6,17E-10	25	5,65E-07 5,40E-07	35	6,68E-05 6,37E-05	11	1,20E+00 1,40E+00
init_start =6	32	3,82E-11 3,66E-11	36	1,76E-10 1,68E-10	36	3,13E-08 2,98E-08	36	6,25E-05 5,96E-05	11	1,20E+00 1,40E+00
init_start =7	37	4,35E-14 4,22E-14	37	2,06E-11 1,97E-11	25	1,30E-06 1,24E-06	37	8,66E-05 8,26E-05	11	1,20E+00 1,40E+00
init_start =8	38	1,20E-13 1,16E-13	38	2,62E-10 2,50E-10	38	5,42E-08 5,17E-08	38	2,78E-05 2,65E-05	11	1,20E+00 1,40E+00
init_start =9	30	3,95E-15 3,93E-15	39	4,87E-12 4,65E-12	39	3,31E-08 3,16E-08	39	6,72E-06 6,41E-06	11	1,20E+00 1,40E+00
init_start =10	40	3,39E-14 3,28E-14	38	2,65E-10 2,31E-10	40	1,85E-08 1,77E-08	40	5,54E-06 5,29E-06	12	1,20E+00 1,40E+00

Table B.8: Example 20 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r}=0,1$			$\bar{r}=0,2$			$\bar{r}=0,3$			$\bar{r}=0,4$			$\bar{r}=0,5$		
init_start =1	12	2,72E-01	2,93E-01	17	6,12E-11	6,57E-11	14	3,66E-15	4,22E-15	22	1,07E-10	1,13E-10	24	1,99E-09	2,05E-09
init_start =2	11	2,72E-01	2,93E-01	15	4,08E-11	3,94E-11	14	1,55E-15	2,11E-15	22	8,68E-14	9,19E-14	22	4,34E-10	4,46E-10
init_start =3	11	2,72E-01	2,93E-01	22	4,15E-13	4,54E-13	14	1,55E-15	2,11E-15	22	9,04E-14	9,59E-14	17	7,76E-09	8,02E-09
init_start =4	12	2,72E-01	2,93E-01	22	8,67E-13	9,37E-13	14	2,89E-15	3,44E-15	17	2,64E-13	2,80E-13	24	3,52E-12	3,68E-12
init_start =5	12	2,72E-01	2,93E-01	22	3,00E-12	3,19E-12	14	2,89E-15	3,44E-15	19	4,44E-15	4,44E-15	24	7,71E-08	8,07E-08
init_start =6	12	2,72E-01	2,93E-01	22	1,33E-15	1,33E-15	14	2,89E-15	3,44E-15	19	4,44E-15	4,44E-15	24	7,99E-11	8,34E-11
init_start =7	12	2,72E-01	2,93E-01	21	2,66E-15	3,22E-15	14	2,89E-15	3,44E-15	22	1,11E-14	1,22E-14	24	1,70E-12	1,76E-12
init_start =8	12	2,72E-01	2,93E-01	22	3,55E-15	3,77E-15	15	1,78E-15	1,55E-15	22	2,11E-15	2,89E-15	24	3,13E-12	3,27E-12
init_start =9	12	2,72E-01	2,93E-01	22	3,55E-15	3,77E-15	15	2,00E-15	2,44E-15	21	4,55E-15	5,77E-15	24	3,13E-12	3,27E-12
init_start =10	12	2,72E-01	2,93E-01	22	1,89E-15	2,22E-15	15	2,00E-15	2,44E-15	21	4,55E-15	5,77E-15	24	2,59E-12	2,70E-12
	$\bar{r}=0,6$			$\bar{r}=0,7$			$\bar{r}=0,8$			$\bar{r}=0,9$			$\bar{r}=1$		
init_start =1	4	8,81E-02	8,66E-02	2	5,86E-01	7,13E-01	2	6,88E-01	7,97E-01	5	2,31E-01	2,32E-01	3	8,92E-01	9,68E-01
init_start =2	4	1,82E-01	2,46E-01							5	2,31E-01	2,32E-01	4	8,92E-01	9,68E-01
init_start =3	4	1,82E-01	2,46E-01							5	5,48E-01	6,67E-01	5	8,92E-01	9,68E-01
init_start =4										5	5,48E-01	6,67E-01	6	8,92E-01	9,68E-01
init_start =5													7	8,92E-01	9,68E-01
init_start =6													8	8,92E-01	9,68E-01
init_start =7													9	8,92E-01	9,68E-01
init_start =8													10	8,92E-01	9,68E-01
init_start =9													11	8,92E-01	9,68E-01
init_start =10													12	8,92E-01	9,68E-01

Table B.9: Example 21 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)}$ random.

	$\bar{r}=0,1$			$\bar{r}=0,2$			$\bar{r}=0,3$			$\bar{r}=0,4$			$\bar{r}=0,5$		
init_start =1	10	2,80E-01	2,96E-01	23	2,33E-14	2,55E-14	16	3,69E-13	4,03E-13	22	1,37E-11	1,48E-11	22	6,20E-11	6,22E-11
init_start =2	10	2,80E-01	2,96E-01	22	1,11E-15	1,11E-15	14	2,11E-15	2,00E-15	22	5,64E-14	6,08E-14	22	3,23E-12	3,25E-12
init_start =3	10	2,80E-01	2,96E-01	23	8,88E-16	1,11E-15	14	2,11E-15	2,00E-15	18	3,12E-13	3,17E-13	17	1,67E-09	1,65E-09
init_start =4	10	2,80E-01	2,96E-01	23	2,62E-14	2,84E-14	14	2,22E-15	2,44E-15	22	2,06E-12	2,22E-12	22	9,32E-12	9,35E-12
init_start =5	10	2,80E-01	2,96E-01	24	3,34E-13	3,62E-13	14	2,22E-15	2,44E-15	22	5,33E-14	5,73E-14	22	2,02E-10	2,03E-10
init_start =6	10	2,80E-01	2,96E-01	22	1,33E-15	1,33E-15	14	2,22E-15	2,44E-15	20	2,66E-15	2,66E-15	22	3,63E-10	3,65E-10
init_start =7	10	2,80E-01	2,96E-01	21	1,78E-15	2,00E-15	14	2,22E-15	2,44E-15	19	3,55E-15	3,77E-15	22	2,59E-10	2,60E-10
init_start =8	10	2,80E-01	2,96E-01	23	2,11E-15	2,00E-15	15	1,89E-15	1,33E-15	19	3,55E-15	3,77E-15	22	9,74E-12	9,78E-12
init_start =9	11	2,80E-01	2,96E-01	23	1,33E-15	1,78E-15	15	1,89E-15	1,33E-15	20	2,55E-15	2,22E-15	22	9,74E-12	9,78E-12
init_start =10	12	2,80E-01	2,96E-01	22	1,33E-15	1,33E-15	15	1,89E-15	1,33E-15	20	2,66E-15	2,66E-15	22	1,06E-11	1,06E-11
	$\bar{r}=0,6$			$\bar{r}=0,7$			$\bar{r}=0,8$			$\bar{r}=0,9$			$\bar{r}=1$		
init_start =1	20	4,91E-09	4,89E-09	22	1,19E-07	1,24E-07	33	3,44E-06	3,65E-06	47	1,31E-06	1,31E-06	3	8,92E-01	9,68E-01
init_start =2	35	1,57E-11	1,57E-11	26	8,10E-09	8,02E-09	40	9,00E-07	9,59E-07	47	1,31E-06	1,31E-06	4	8,92E-01	9,68E-01
init_start =3	35	4,30E-12	4,27E-12	21	9,43E-07	1,01E-06	53	7,42E-09	7,47E-09	47	1,84E-05	1,94E-05	6	8,92E-01	9,68E-01
init_start =4	35	5,67E-12	5,69E-12	31	1,39E-06	1,39E-06	49	1,45E-09	1,44E-09	47	1,84E-05	1,94E-05	6	8,92E-01	9,68E-01
init_start =5	35	1,73E-11	1,73E-11	29	2,95E-07	2,93E-07	46	3,20E+13	3,19E+13	47	1,21E-05	1,28E-05	7	8,92E-01	9,68E-01
init_start =6	35	3,58E-13	3,58E-13	40	1,04E-11	1,04E-11	31	1,54E+12	1,63E+12	47	1,21E-05	1,28E-05	8	8,92E-01	9,68E-01
init_start =7	35	5,66E-13	5,67E-13	44	3,66E-08	3,64E-08	31	1,49E+10	1,53E+10	47	8,86E-06	8,74E-06	9	8,92E-01	9,68E-01
init_start =8	35	6,06E-12	6,05E-12	40	1,25E-11	1,24E-11	49	1,43E-09	1,41E-09	47	8,86E-06	8,74E-06	11	8,92E-01	9,68E-01
init_start =9	35	1,62E-12	1,63E-12	46	1,15E-11	1,16E-11	49	1,42E-09	1,41E-09	47	1,13E-05	1,18E-05	11	8,92E-01	9,68E-01
init_start =10	44	1,58E-12	1,71E-12	28	1,45E+10	1,51E+10	44	2,66E-08	2,78E-08	47	1,13E-05	1,18E-05	12	8,92E-01	9,68E-01

Table B.10: Example 21 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)}$ random.

	$\bar{r} = 0,1$	$\bar{r} = 0,2$	$\bar{r} = 0,3$	$\bar{r} = 0,4$	$\bar{r} = 0,5$
init_start =1				2 1,08E+00 4,31E-01	
init_start =2					
init_start =3					
init_start =4					
init_start =5					
init_start =6					
init_start =7					
init_start =8					
init_start =9					
init_start =10					

	$\bar{r} = 0,6$	$\bar{r} = 0,7$	$\bar{r} = 0,8$	$\bar{r} = 0,9$	$\bar{r} = 1$
init_start =1			5 5,15E-01 4,26E-01		3 3,69E+00 1,34E+00
init_start =2			5 5,15E-01 4,26E-01		4 3,69E+00 1,34E+00
init_start =3			5 1,19E+00 4,03E-01		5 3,69E+00 1,34E+00
init_start =4					6 3,69E+00 1,34E+00
init_start =5					7 3,69E+00 1,34E+00
init_start =6					8 3,69E+00 1,34E+00
init_start =7					9 3,69E+00 1,34E+00
init_start =8					10 3,69E+00 1,34E+00
init_start =9					11 3,69E+00 1,34E+00
init_start =10					12 3,69E+00 1,34E+00

Table B.11: Example 22 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r} = 0,1$	$\bar{r} = 0,2$	$\bar{r} = 0,3$	$\bar{r} = 0,4$	$\bar{r} = 0,5$
init_start =1	8 1,97E+00 1,68E+00	15 2,35E-02 2,00E-02	21 5,83E-15 7,11E-15	18 5,55E-16 5,55E-16	22 2,00E-15 1,55E-15
init_start =2	8 1,97E+00 1,68E+00	14 2,35E-02 2,00E-02	21 5,72E-13 6,18E-13	18 6,66E-16 5,55E-16	22 1,39E-15 1,11E-15
init_start =3	8 1,97E+00 1,68E+00	15 2,35E-02 2,00E-02	20 3,94E-15 3,55E-15	17 4,44E-15 4,27E-15	22 1,39E-15 1,11E-15
init_start =4	8 1,97E+00 1,68E+00	15 2,35E-02 2,00E-02	20 3,94E-15 3,55E-15	17 4,44E-15 4,27E-15	20 2,44E-15 2,50E-15
init_start =5	9 1,97E+00 1,68E+00	15 2,35E-02 2,00E-02	20 3,94E-15 3,55E-15	17 1,89E-15 1,67E-15	19 4,61E-15 4,66E-15
init_start =6	11 1,97E+00 1,68E+00	15 2,35E-02 2,00E-02	21 1,55E-15 1,11E-15	17 1,89E-15 1,67E-15	18 4,83E-15 4,94E-15
init_start =7	11 1,97E+00 1,68E+00	15 2,35E-02 2,00E-02	21 3,66E-15 3,89E-15	17 4,44E-15 4,27E-15	18 4,83E-15 4,94E-15
init_start =8	11 1,97E+00 1,68E+00	15 2,35E-02 2,00E-02	21 3,66E-15 3,89E-15	17 4,44E-15 4,27E-15	16 3,28E-15 3,22E-15
init_start =9	11 1,97E+00 1,68E+00	15 2,35E-02 2,00E-02	21 2,61E-15 1,94E-15	18 6,66E-16 5,55E-16	15 1,61E-15 1,33E-15
init_start =10	13 1,97E+00 1,68E+00	16 2,35E-02 2,00E-02	21 2,61E-15 1,94E-15	18 8,60E-16 7,77E-16	15 1,61E-15 1,33E-15

	$\bar{r} = 0,6$	$\bar{r} = 0,7$	$\bar{r} = 0,8$	$\bar{r} = 0,9$	$\bar{r} = 1$
init_start =1	38 1,20E-14 1,21E-14	33 2,51E-05 1,68E-05	51 2,61E-05 1,72E-05	51 1,26E-01 1,08E-01	3 3,69E+00 1,34E+00
init_start =2	26 1,72E-15 1,22E-15	33 2,51E-05 1,68E-05	52 3,53E-05 2,60E-05	52 8,63E-02 7,34E-02	4 3,69E+00 1,34E+00
init_start =3	26 1,72E-15 1,22E-15	33 5,64E-08 3,55E-08	53 5,58E-05 4,06E-05	53 6,80E-02 5,78E-02	5 3,69E+00 1,34E+00
init_start =4	26 1,72E-15 1,22E-15	33 5,64E-08 3,55E-08	54 3,36E-05 2,40E-05	54 6,58E-02 5,60E-02	6 3,69E+00 1,34E+00
init_start =5	26 1,72E-15 1,22E-15	33 9,29E-11 5,49E-11	55 2,49E-07 1,77E-07	55 6,11E-02 5,20E-02	7 3,69E+00 1,34E+00
init_start =6	38 4,88E-14 4,77E-14	33 9,29E-11 5,49E-11	56 8,97E-09 6,25E-09	56 6,16E-02 5,24E-02	8 3,69E+00 1,34E+00
init_start =7	33 4,55E-15 4,44E-15	33 3,13E-12 1,43E-12	57 1,11E-09 7,47E-10	57 7,08E-02 6,02E-02	9 3,69E+00 1,34E+00
init_start =8	32 2,55E-15 2,22E-15	33 3,13E-12 1,43E-12	58 1,19E-10 7,68E-11	58 6,69E-02 5,69E-02	10 3,69E+00 1,34E+00
init_start =9	38 1,13E-14 1,10E-14	33 3,61E-13 2,07E-13	59 8,50E-11 5,27E-11	59 6,74E-02 5,74E-02	11 3,69E+00 1,34E+00
init_start =10	27 1,11E-15 1,22E-15	33 3,61E-13 2,07E-13	57 1,07E-10 6,65E-11	60 6,70E-02 5,70E-02	12 3,69E+00 1,34E+00

Table B.12: Example 22 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r} = 0,1$			$\bar{r} = 0,2$			$\bar{r} = 0,3$			$\bar{r} = 0,4$			$\bar{r} = 0,5$		
init_start =1				16	2,24E-05	9,15E-04				31	4,20E-07	8,92E-04			
init_start =2				16	2,24E-05	9,15E-04				28	1,74E+07	1,82E+07			
init_start =3				16	2,24E-05	9,15E-04				31	4,97E-07	8,92E-04			
init_start =4				16	2,24E-05	9,15E-04				31	4,51E-07	8,92E-04			
init_start =5				16	2,24E-05	9,15E-04				31	4,46E-07	8,92E-04			
init_start =6				16	2,24E-05	9,15E-04				21	1,44E+07	1,51E+07			
init_start =7				16	2,24E-05	9,15E-04				31	4,89E-07	8,92E-04			
init_start =8				16	2,24E-05	9,15E-04				31	4,84E-07	8,92E-04			
init_start =9				16	2,24E-05	9,15E-04				31	4,95E-07	8,92E-04			
init_start =10				16	2,24E-05	9,15E-04				31	4,95E-07	8,92E-04			
	$\bar{r} = 0,6$			$\bar{r} = 0,7$			$\bar{r} = 0,8$			$\bar{r} = 0,9$			$\bar{r} = 1$		
init_start =2	2	7,13E-01	7,05E-01	2	8,26E-01	8,17E-01	31	2,59E-07	8,92E-04	2	1,05E+00	1,04E+00	3	1,16E+00	1,15E+00
init_start =3							32	2,46E-07	8,92E-04				4	1,16E+00	1,15E+00
init_start =4							33	4,28E-08	8,92E-04				5	1,16E+00	1,15E+00
init_start =5							34	3,49E-07	8,92E-04				6	1,16E+00	1,15E+00
init_start =6							35	2,61E-07	8,92E-04				7	1,16E+00	1,15E+00
init_start =7							36	1,60E-07	8,92E-04				8	1,16E+00	1,15E+00
init_start =8							37	3,63E-07	8,92E-04				9	1,16E+00	1,15E+00
init_start =9							38	8,25E-08	8,92E-04				10	1,16E+00	1,15E+00
init_start =10							39	3,54E-07	8,92E-04				11	1,16E+00	1,15E+00
							40	2,56E-07	8,92E-04				12	1,16E+00	1,15E+00

Table B.13: Example 23 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r} = 0,1$			$\bar{r} = 0,2$			$\bar{r} = 0,3$			$\bar{r} = 0,4$			$\bar{r} = 0,5$		
init_start =1	12	2,73E-05	9,20E-04	20	5,81E-07	8,93E-04	24	1,06E-08	8,92E-04	31	1,58E-08	8,92E-04	31	1,33E-10	8,92E-04
init_start =2	12	2,73E-05	9,20E-04	15	1,28E+10	1,32E+10	24	1,56E-08	8,92E-04	30	2,28E-09	8,92E-04	24	1,72E+11	1,78E+11
init_start =3	12	2,73E-05	9,20E-04	20	5,79E-07	8,93E-04	24	1,96E-08	8,92E-04	18	2,73E+12	2,82E+12	26	2,64E-09	8,92E-04
init_start =4	12	2,73E-05	9,20E-04	20	5,82E-07	8,93E-04	24	2,42E-08	8,92E-04	34	2,56E-09	8,92E-04	31	4,04E-09	8,92E-04
init_start =5	14	2,73E-05	9,20E-04	20	5,82E-07	8,93E-04	24	2,18E-08	8,92E-04	35	1,85E-09	8,92E-04	35	1,20E-10	8,92E-04
init_start =6	14	2,73E-05	9,20E-04	20	5,82E-07	8,93E-04	24	2,20E-08	8,92E-04	36	1,08E-09	8,92E-04	36	9,53E-11	8,92E-04
init_start =7	14	2,73E-05	9,20E-04	20	5,82E-07	8,93E-04	24	2,31E-08	8,92E-04	36	1,85E-08	8,92E-04	37	8,40E-11	8,92E-04
init_start =8	14	2,73E-05	9,20E-04	20	5,82E-07	8,93E-04	24	2,79E-08	8,92E-04	36	2,06E-09	8,92E-04	38	1,66E-09	8,92E-04
init_start =9	14	2,73E-05	9,20E-04	22	5,82E-07	8,93E-04	24	2,62E-08	8,92E-04	25	1,38E+10	1,43E+10	39	1,93E-10	8,92E-04
init_start =10	14	2,73E-05	9,20E-04	22	5,82E-07	8,93E-04	24	2,47E-08	8,92E-04	27	1,22E-09	8,92E-04	40	9,97E-12	8,92E-04
	$\bar{r} = 0,6$			$\bar{r} = 0,7$			$\bar{r} = 0,8$			$\bar{r} = 0,9$			$\bar{r} = 1$		
init_start =1	31	3,25E-10	8,92E-04	31	4,53E-08	8,92E-04	31	3,74E-07	8,92E-04	31	6,40E-06	8,99E-04	3	1,16E+00	1,15E+00
init_start =2	32	5,22E-08	8,92E-04	32	1,40E-08	8,92E-04	32	2,14E-07	8,92E-04	32	6,89E-06	8,99E-04	5	1,16E+00	1,15E+00
init_start =3	30	9,43E-07	8,93E-04	33	1,81E-07	8,92E-04	33	1,67E-07	8,92E-04	33	7,27E-06	8,99E-04	5	1,16E+00	1,15E+00
init_start =4	34	6,13E-08	8,92E-04	34	1,24E-06	8,91E-04	34	1,21E-07	8,92E-04	34	7,18E-06	8,99E-04	6	1,16E+00	1,15E+00
init_start =5	35	1,05E-10	8,92E-04	35	1,02E-06	8,93E-04	35	1,21E-07	8,92E-04	35	6,61E-06	8,99E-04	7	1,16E+00	1,15E+00
init_start =6	36	1,37E-10	8,92E-04	36	8,38E-06	8,83E-04	36	1,48E-06	8,93E-04	36	6,54E-06	8,99E-04	8	1,16E+00	1,15E+00
init_start =7	37	6,83E-11	8,92E-04	37	1,99E-09	8,92E-04	37	1,27E-07	8,92E-04	37	4,93E-05	9,43E-04	9	1,16E+00	1,15E+00
init_start =8	38	1,56E-10	8,92E-04	38	9,34E-09	8,92E-04	38	5,81E-07	8,91E-04	38	6,80E-04	3,03E-04	12	1,16E+00	1,15E+00
init_start =9	39	1,72E-11	8,92E-04	39	1,37E-08	8,92E-04	39	2,98E-07	8,92E-04	39	1,41E-04	1,04E-03	12	1,16E+00	1,15E+00
init_start =10	40	2,19E-11	8,92E-04	40	1,53E-09	8,92E-04	40	3,58E-06	8,88E-04	40	1,97E-05	9,12E-04	12	1,16E+00	1,15E+00

Table B.14: Example 23 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r}=0,1$			$\bar{r}=0,2$			$\bar{r}=0,3$			$\bar{r}=0,4$			$\bar{r}=0,5$		
init_start =1	9	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	14	1,27E-01	1,65E-01						
init_start =2	9	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	14	1,27E-01	1,65E-01						
init_start =3	9	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	14	1,27E-01	1,65E-01						
init_start =4	9	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	14	1,27E-01	1,65E-01						
init_start =5	9	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	14	1,27E-01	1,65E-01						
init_start =6	15	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	14	1,27E-01	1,65E-01						
init_start =7	15	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	15	1,27E-01	1,65E-01						
init_start =8	15	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	14	1,27E-01	1,65E-01						
init_start =9	15	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	15	1,27E-01	1,65E-01						
init_start =10	15	2,47E-01	3,38E-01	12	1,75E-01	2,33E-01	15	1,27E-01	1,65E-01						
	$\bar{r}=0,6$			$\bar{r}=0,7$			$\bar{r}=0,8$			$\bar{r}=0,9$			$\bar{r}=1$		
init_start =1										2	9,30E-01	0,87E-01	3	1	9,36E-01
init_start =2													4	1	9,36E-01
init_start =3													5	1	9,36E-01
init_start =4													6	1	9,36E-01
init_start =5													7	1	9,36E-01
init_start =6													8	1	9,36E-01
init_start =7													9	1	9,36E-01
init_start =8													10	1	9,36E-01
init_start =9													11	1	9,36E-01
init_start =10													12	1	9,36E-01

Table B.15: Example 24 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} \equiv 1$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

	$\bar{r}=0,1$			$\bar{r}=0,2$			$\bar{r}=0,3$			$\bar{r}=0,4$			$\bar{r}=0,5$		
init_start =1	12	1,23E-02	7,54E-03	18	2,70E-09	3,92E-09	18	9,07E-08	8,80E-08	25	2,45E-08	3,11E-08	27	1,11E-04	1,11E-04
init_start =2	12	1,23E-02	7,54E-03	18	2,54E-09	9,16E-09	22	2,00E-09	8,62E-09	23	2,06E-08	2,73E-08	28	6,63E-07	6,61E-07
init_start =3	12	1,23E-02	7,54E-03	18	2,54E-09	9,16E-09	22	3,56E-10	6,98E-09	29	9,50E-10	7,57E-09	28	5,00E-07	4,97E-07
init_start =4	12	1,23E-02	7,54E-03	18	1,47E-10	6,77E-09	22	1,12E-09	7,74E-09	29	1,50E-07	1,48E-07	28	5,61E-09	2,88E-09
init_start =5	12	1,23E-02	7,54E-03	18	1,47E-10	6,77E-09	22	4,55E-09	1,12E-08	29	1,18E-09	7,81E-09	28	4,58E-06	4,58E-06
init_start =6	12	1,23E-02	7,54E-03	18	1,69E-10	6,79E-09	19	8,64E-09	6,34E-09	29	3,72E-08	6,26E-08	26	3,85E-06	3,85E-06
init_start =7	13	1,23E-02	7,54E-03	18	1,69E-10	6,79E-09	22	1,35E-09	7,97E-09	29	7,52E-09	1,41E-08	21	1,92E-05	1,92E-05
init_start =8	13	1,23E-02	7,54E-03	18	1,19E-10	6,50E-09	22	7,19E-10	5,90E-09	29	5,48E-10	7,17E-09	28	1,65E-08	2,31E-08
init_start =9	13	1,23E-02	7,54E-03	18	1,21E-10	6,74E-09	22	7,19E-10	5,90E-09	29	3,93E-10	7,02E-09	28	4,29E-03	4,29E-03
init_start =10	14	1,23E-02	7,54E-03	18	1,21E-10	6,74E-09	22	3,92E-11	6,66E-09	29	3,93E-10	7,02E-09	28	1,03E-08	1,69E-08
	$\bar{r}=0,6$			$\bar{r}=0,7$			$\bar{r}=0,8$			$\bar{r}=0,9$			$\bar{r}=1$		
init_start =1	31	4,36E-06	4,37E-06	31	9,02E-06	9,03E-06	31	2,65E-04	2,68E-04	31	7,27E-03	7,45E-03	3	1,00E+00	9,36E-01
init_start =2	32	2,09E-04	2,09E-04	32	6,23E-06	6,23E-06	32	1,69E-04	1,67E-04	32	7,17E-03	7,37E-03	4	1,00E+00	9,36E-01
init_start =3	33	2,41E-05	2,41E-05	33	6,60E-06	6,61E-06	33	1,27E-04	1,26E-04	33	7,19E-03	7,39E-03	5	1,00E+00	9,36E-01
init_start =4	33	2,51E-07	2,51E-07	34	4,41E-06	4,42E-06	34	1,26E-04	1,25E-04	34	7,31E-03	7,51E-03	6	1,00E+00	9,36E-01
init_start =5	33	8,97E-08	9,63E-08	35	3,13E-06	3,14E-06	35	1,04E-04	1,04E-04	35	9,52E-03	9,74E-03	7	1,00E+00	9,36E-01
init_start =6	33	7,88E-07	7,85E-07	36	3,01E-06	3,01E-06	36	5,02E-05	5,01E-05	20	7,77E+19	4,62E+01	8	1,00E+00	9,36E-01
init_start =7	33	8,20E-08	8,87E-08	37	5,87E-07	7,98E-07	37	9,63E-05	9,59E-05	20	7,77E+19	4,62E+01	9	1,00E+00	9,36E-01
init_start =8	33	3,54E-08	4,20E-08	38	7,84E-05	7,84E-05	38	1,46E-04	1,45E-04	38	8,71E-03	8,91E-03	10	1,00E+00	9,36E-01
init_start =9	30		Inf 6,48E+15	33	7,12E-06	7,13E-06	39	2,00E-04	1,98E-04	39	5,50E-03	5,66E-03	11	1,00E+00	9,36E-01
init_start =10	30		Inf 6,00E+15	31	1,09E-05	1,09E-05	40	5,16E-05	5,15E-05	40	4,05E-03	4,19E-03	12	1,00E+00	9,36E-01

Table B.16: Example 24 - The number of iterations to which the algorithm has stopped, the residual and error associated to the last extrapolated vector with the dual vector $\mathbf{y} = \mathbf{w}$ and the starting vector $\mathbf{u}^{(0)} \equiv 1$.

Appendix C

MATLAB software

In the following frame we report the MATLAB code of AM_eqint algorithm that we used in order to establish the results of this thesis.

```
%AM_eqint.m
% p:          the number of the quadrature formula nodes minus 1
% w:          the vector of the quadrature formula weights
% no:         the matrix of the quadrature formula nodes
% zeta:       the zeta value that descrived the confidential region
% dim:        the dimension of domain
% norma:      the norm use for measure the residuals and the errors
% U_start:    the starting vector
% solfun:     the handle function corresponds to the exact solution
% sol:        the evalutation of solfun on quadrature nodes
% ifun:       the handle function corresponds to the kernel of integral equation
% ifun:       the handle function corresponds to the f(t) function of int. eq.
% NBCI:       the maximum number of iterations
% TOLR:       the tolerance for the stopping criterion on residual
% MAXCOL:     the parameter 2k of extrapolation
% Y:          the dual vector for STEA2
% alch:       the choice of strategy
%             -> 1 for the fixed step size strategy
%             -> 3 for the adaptive step size strategy
% al:         the parameter alpha for the fixed step size strategy
% sec_Tol:    the tolerance of secant method for adaptive step size strategy
% sec_ratio_target: the fixed ratio for adaptive step size strategy
% init_iter: the number of iteration after that we start the extrapolation
%             in the adaptive strategy

% INITIALIZE THE DATA STRUCTURE FOR ERRORS

% Error of Picard iteration respect continuos solution ||u - sol||
nerrU = zeros(init_iter+NBCI,1);
% Error of epsilon_algorithm respect continuos solution ||eps - sol||
nerrEPS = zeros(init_iter+NBCI,1);
% Residual of Picard iteration respect continuos solution ||u - sol||
nerrF = zeros(init_iter+NBCI,1);
% Residual of epsilon_algorithm respect continuos solution ||eps - sol||
nerrFEPS = zeros(init_iter+NBCI,1);
% Residual
nres = zeros(init_iter+NBCI,p+1);
nratios = zeros(init_iter+NBCI);
NBCt = NBCI;

```

```
41 %INITIALIZE THE DATA STRUCTURE FOR APPROXIMATE SOLUTION
42 U_AS = zeros(p+1,1);
43 EPSINI_AS = zeros(p+1,1);
44
45 %INITIALIZE THE DATA STRUCTURE FOR STORAGE THE ALPHA'S VALUES
46 alpha_history = zeros(init_iter+NBCI,1);
47
48 if init_iter > 0
49     U = U_start;
50     for k=1:init_iter
51
52         %BEGIN[Save the errors]
53         nerrU(k)    = norma(sol-U);
54         F = T_eqint(U,no,w,ifun,funt,dim);
55         nres(k,:) = U - F; nerrF(k) = norma(nres(k,:));
56         nratios(k,:) = Y'*(U-sol);
57         % In F there is F(U)
58         %END[Save the errors]
59
60         %sec_TU = T_eqint(U,no,w,ifun,funt,dim);
61         sec_TU = F;
62         sec_diff_U = U-sec_TU;
63         sec_val_U = Y'*(sec_diff_U);
64         sec_target_prod = sec_val_U*sec_ratio_target;
65         %
66         al_pre = 0.95;
67         sec_sol_temp = (1-al_pre)*U+al_pre*sec_TU;
68         sec_val_pre = Y'*(sec_sol_temp-T_eqint(sec_sol_temp,no,w,ifun,funt,dim));
69         %
70         al_cor = 1;
71         sec_sol_temp = (1-al_cor)*U+al_cor*sec_TU;
72         sec_val_cor = Y'*(sec_sol_temp-T_eqint(sec_sol_temp,no,w,ifun,funt,dim));
73         %
74         while abs(sec_val_cor-sec_target_prod)>sec_Tol
75             al_suc = al_cor-(sec_val_cor-sec_target_prod)*(al_cor-al_pre)/...
76                 (sec_val_cor-sec_val_pre);
77             al_pre = al_cor; al_cor = al_suc;
78             sec_val_pre = sec_val_cor;
79             sec_sol_temp = (1-al_cor)*U+al_cor*sec_TU;
80             sec_val_cor = Y'*(sec_sol_temp-T_eqint(sec_sol_temp,no,w,ifun,funt,dim));
81         end
82         alpha_history(k+1)=al_cor;
83         U = U-al_cor*(U-sec_TU);
84
85     end
86     U_start = U;
87 end
88 % Apply STEA2 to theexample chosen
89
90     % Arguments and initializations for the first call of the SEAW and STEA
91
92 U = U_start;
93 % First value <y,x_0>
94
95 estr = STEA2(NBCI,MAXCOL,Y, true);
96 estr.insert_new_vector(U);
97 EPSINI = estr.extrapolation();
98
99 %BEGIN[Save the errors]
100 nerrU(1+init_iter)    = norma(sol-U);
101 nerrEPS(1+init_iter) = norma(sol-EPSINI);
102 F = T_eqint(U,no,w,ifun,funt,dim);
103 nres(1+init_iter,:) = U - F; nerrF(1+init_iter) = norma(nres(1+init_iter,:));
104 nratios(1+init_iter,:) = Y'*(U-sol);
105 nerrFEPS(1+init_iter) = norma(EPSINI -T_eqint(EPSINI,no,w,ifun,funt,dim));
```

```
106 % In F there is F(U)
107 %END[Save the errors]
108
109 % In the first iteration we consider the initial term
110 disp('Iteration 1 completed (starting vector)');
111
112 % Start of the loop
113 for ii = 2:NBCI
114     i = ii + init_iter;
115     % Determinate the new Picard iteration
116     if alch==1
117         sec_TU = T_eqint(U,no,w,ifun,funt,dim);
118         sec_diff_U = U-sec_TU;
119         sec_val_U = Y'*(sec_diff_U);
120         sec_target_prod = sec_val_U*sec_ratio_target;
121         %
122         al_pre = 0.95;
123         sec_sol_temp = (1-al_pre)*U+al_pre*sec_TU;
124         sec_val_pre = Y'*(sec_sol_temp-T_eqint(sec_sol_temp,no,w,ifun,funt,dim));
125         %
126         al_cor = 1;
127         sec_sol_temp = (1-al_cor)*U+al_cor*sec_TU;
128         sec_val_cor = Y'*(sec_sol_temp-T_eqint(sec_sol_temp,no,w,ifun,funt,dim));
129         %
130         while abs(sec_val_cor-sec_target_prod)>sec_Tol
131             al_suc = al_cor-(sec_val_cor-sec_target_prod)*(al_cor-al_pre)/...
132                 (sec_val_cor-sec_val_pre);
133             al_pre = al_cor; al_cor = al_suc;
134             sec_val_pre = sec_val_cor;
135             sec_sol_temp = (1-al_cor)*U+al_cor*sec_TU;
136             sec_val_cor = Y'*(sec_sol_temp-T_eqint(sec_sol_temp,no,w,ifun,funt,dim));
137         end
138         if sec_debug>0
139             disp(['Residual alpha = ',num2str(abs(sec_val_cor-sec_target_prod),'%6.2e')]);
140             disp(['Error alpha = ',num2str(abs(al_cor-al_th),'%6.2e')]);
141         end
142         al = al_cor;
143     end
144     alpha_history(i)=al;
145     U = U-al*(U-F);
146
147     %Check current solution is in [-zeta,zeta]
148     if norm(U,Inf)>zeta
149         warning('the current solution is outside of the conditions');
150     end
151     % New value of <y,U>
152     estr.insert_new_vector(U);
153     EPSINI = estr.extrapolation();
154
155     disp(['Iteration ', num2str(i,'%d'), ' completed']);
156
157     %BEGIN[Save the errors]
158     nerrU(i)   = norma(sol-U);
159     nerrEPS(i) = norma(sol-EPSINI);
160     F = T_eqint(U,no,w,ifun,funt,dim);
161     nres(i,:) = U - F; nerrF(i) = norma(nres(i,:));
162     nerrFEPS(i) = norma(EPSINI -T_eqint(EPSINI,no,w,ifun,funt,dim));
163     nratios(i,:) = Y'*(U-sol);
164     % In F there is F(U)
165     %END[Save the errors]
166     if nerrFEPS(i) <= TOLR
167         disp([' STOPPED at iteration ',num2str(i)]);
168         break
169     end
170 end
```

```
%T_eqint.m
function T = T_eqint(U,no,w,ifun,funt,dim)
% U:          the current solution of method
% no:         the matrix of the quadrature formula nodes
% w:          the vector of the quadrature formula weights
% ifun:       the handle function corresponds to the f(t) function of int. eq.
% ifun:       the handle function corresponds to the kernel of integral equation
% dim:        the dimension of domain
if (nargin < 6) || (nargin > 6)
    error(' T_eqint: Not enought or too many input parameters')
end
if dim==1
    m = length(no);

    T = funt(no);
    for j=1:m
        T(1:m) = T(1:m)+w(j)*ifun(no(1:m),no(j),U(j));
    end
elseif dim==2
    T = zeros(size(no,1),1);
    for i=1:size(no,1)
        T(i) = funt(no(i,1),no(i,2));
        for j=1:size(no,1)
            T(i) = T(i) + w(j)*ifun(no(i,1),no(i,2),no(j,1),no(j,2),U(j));
        end
    end
end
```

In the following frames we show the MATLAB code of our implementation of simplified ε-algorithm.

```
% SEA_sing_class.m
classdef SEA_sing_class < handle
    %Scalar Epsilon-Algorithm
    properties (Access = private)
        is_closed  % flag active if we have established all singularity
                   % first column of singularity
        m          % order of singularity
        current_m  % index of current singularity
        W          % vector of W elements
        alpha      % vector of alpha elements
        N          % vector of N elements
        C          % vector of C elements (determinated from W,alpha)
    end

    methods
        function obj = SEA_sing_class(alpha1, preallocate)
            if nargin==1
                preallocate = 10;
            end
            obj.is_closed=false;
            obj.m = 0;
            obj.W = zeros(1,preallocate);
            obj.N = zeros(1,preallocate);
            obj.C = zeros(1,preallocate);
            obj.alpha = zeros(1,preallocate+1);
            obj.alpha(1) = alpha1;
        end

        function ind = storeWalpha(obj,W,alpha)
            %store W, alpha, N and C
            %ok
            obj.m = obj.m + 1;
```

```
33              obj.W(obj.m) = W;
34              obj.alpha(obj.m+1) = alpha;
35              obj.C(obj.m) = W+1/(alpha-obj.alpha(obj.m));
36              ind = obj.m*2;
37          end
38
39          function storeN(obj,N)
40              obj.N(obj.m) = N;
41          end
42
43          function isclosedbool = isclosed(obj)
44              isclosedbool = obj.is_closed;
45          end
46
47          function close(obj)
48              obj.is_closed = true;
49              obj.current_m = obj.m;
50          end
51
52          function [L,E] = Cordelier(obj,S)
53              my_m = obj.current_m;
54              obj.current_m = my_m - 1;
55              if my_m > 0
56                  r = S/(1-S/obj.C(my_m)) +...
57                      obj.N(my_m)/(1-obj.N(my_m)/obj.C(my_m)) - ...
58                      obj.W(my_m)/(1-obj.W(my_m)/obj.C(my_m));
59                  E = r /(1 + r/obj.C(my_m));
60                  L = 2*my_m;
61              else
62                  L = 0;
63                  E = NaN;
64              end
65          end
66
67      end
68  end
```

```
1   %SEA.m
2   classdef SEA < handle
3       %Scalar Epsilon-Algorithm
4       properties (Access = private)
5           diag          % matrix where last two diagonals are stored
6           c_ind         % index of row in diag matrix for the current diag
7           l_ind         % index of row in diag matrix for the last diag
8           p             % parameter p for determinate singularity
9           lastrow       % index of last row in matrix used
10          col           % number of real column
11          % (+1 if we consider the first column of zeros)
12          debug         % if true, we save the diagonal in eps
13          eps           % matrix where Wynn-coefficients are stored (if debug)
14          sing          % list of singularity
15          n_sing        % number of singularity
16          sing_col      % index of sing <list> for every column in current
17          % iteration
18      end
19
20      methods
21          function obj = SEA(N,k,debug)
22              if nargin == 2
23                  debug = false;
24              end
25              if debug
26                  obj.eps = zeros(N+1,k+2);
```

```
27                  obj.debug=true;
28              else
29                  obj.debug=false;
30              end
31              obj.p = -8;
32
33              obj.c_ind = 2; obj.l_ind = 1;
34              obj.diag = zeros(2,k+2);
35              obj.lastrow = 0;
36              obj.col = k+2;
37
38              obj.sing = SEA_sing_class.empty(10,0); %da approfondire
39              obj.n_sing = 0;
40              obj.sing_col = zeros(1,obj.col);
41          end
42
43          function insert_new_S(obj,s)
44              obj.l_ind = obj.c_ind; obj.c_ind = 3 - obj.c_ind;
45              obj.lastrow = obj.lastrow +1;
46              n = min(obj.lastrow+1,obj.col);
47
48              k = 2;
49              while k<=n
50                  if k==2
51                      obj.diag(obj.c_ind,2) = s;
52                  else
53                      obj.diag(obj.c_ind,k) = obj.diag(obj.l_ind,k-2)+...
54                          1/(obj.diag(obj.c_ind,k-1)-obj.diag(obj.l_ind,k-1));
55                  end
56                  %extension if Wynn's particolar rules from Cordelier formula
57                  if obj.sing_col(k)>0
58                      if obj.sing(obj.sing_col(k)).isclosed()
59                          %%% START BLOCK 1
60                          ind_curr_sing = obj.sing_col(k);
61                          obj.sing_col(k) = 0; %set index to zero of k column
62                          %
63                          k = k+1;
64                          Sm = obj.diag(obj.l_ind,k-2)+1/...
65                              (obj.diag(obj.c_ind,k-1)-obj.diag(obj.l_ind,k-1));
66                          obj.diag(obj.c_ind,k) = Sm;
67                          obj.sing_col(k) = 0;
68                          %
69                          [L,E] = obj.sing(ind_curr_sing)...
70                              .Cordelier(obj.diag(obj.c_ind,k));
71                          %set index to zero of k+1 column
72                          if L>0
73                              l = 1;
74                              while l < L && k<n
75                                  k = k+1;
76                                  obj.diag(obj.c_ind,k) = ...
77                                      obj.diag(obj.l_ind,k-2)+...
78                                      1/(obj.diag(obj.c_ind,k-1)-...
79                                      obj.diag(obj.l_ind,k-1));
80                                  l = l+1;
81                              end
82                              k = k+1;
83                              obj.diag(obj.c_ind,k) = E;
84                          end
85                          %%% END BLOCK 1
86                      else
87                          if abs(obj.diag(obj.c_ind,k)-obj.diag(obj.l_ind,k)) ...
88                                  < 10^obj.p * abs(obj.diag(obj.l_ind,k))
89                              %%% START BLOCk 2
90                              index_of_current_singularity = obj.sing_col(k);
91                              L = obj.sing(obj.sing_col(k))...
```

```
92                                  .storeWalpha(obj.diag(obj.l_ind,k-1),...
93                                  obj.diag(obj.c_ind,k));
94                              l = 1;
95                              temp_k=-1;
96                              while l <= L && k<n
97                                  k = k + 1;
98                                  if obj.sing_col(k)==0
99                                      obj.sing_col(k) = index_of_current_singularity;
100                                 elseif temp_k==-1 && obj.sing_col(k) ~= ...
101                                         index_of_current_singularity
102                                     temp_k = k - 1;
103                                 end
104                                 obj.diag(obj.c_ind,k) = obj.diag(obj.l_ind,k-2)+...
105                                     1/(obj.diag(obj.c_ind,k-1)-...
106                                     obj.diag(obj.l_ind,k-1));
107                                 l = l + 1;
108                             end
109                             %clear pre_diag
110                             obj.sing(obj.sing_col(k)).storeN(obj.diag(obj.l_ind,k-1));
111                             if temp_k > -1
112                                 k = temp_k;
113                             end
114                             %%% END BLOCk 2
115                         else
116                             %chiudiamo la singolarità
117                             obj.sing(obj.sing_col(k)).close();
118                             %%% START BLOCK 1
119                             ind_curr_sing = obj.sing_col(k);
120                             obj.sing_col(k) = 0; %set index to zero of k column
121                             %
122                             k = k + 1;
123                             Sm = obj.diag(obj.l_ind,k-2)+1/...
124                                 (obj.diag(obj.c_ind,k-1)-obj.diag(obj.l_ind,k-1));
125                             obj.diag(obj.c_ind,k) = Sm;
126                             obj.sing_col(k) = 0;  %set index to zero of k+1 column
127                             %
128                             [L,E] = obj.sing(ind_curr_sing)...
129                                 .Cordelier(obj.diag(obj.c_ind,k));
130                             %set index to zero of k+1 column
131                             if L>0
132                                 l = 1;
133                                 while l < L && k<n
134                                     k = k+1;
135                                     obj.diag(obj.c_ind,k) = obj.diag(obj.l_ind,k-2)+...
136                                         1/(obj.diag(obj.c_ind,k-1)-...
137                                         obj.diag(obj.l_ind,k-1));
138                                     l = l+1;
139                                 end
140                                 k = k+1;
141                                 obj.diag(obj.c_ind,k) = E;
142                             end
143                             %%% END BLOCK 1
144                         end
145                     end
146                 else
147                     if abs(obj.diag(obj.c_ind,k)-obj.diag(obj.l_ind,k)) ...
148                             < 10^obj.p * abs(obj.diag(obj.l_ind,k))
149                         % create singularity element in list and indicize
150                         obj.n_sing = obj.n_sing + 1;
151                         obj.sing(obj.n_sing) = SEA_sing_class(obj.diag(obj.l_ind,k));
152                         obj.sing_col(k) = obj.n_sing;
153                         %%% START BLOCk 2
154                         index_of_current_singularity = obj.sing_col(k);
155                         L = obj.sing(obj.sing_col(k))...
156                             .storeWalpha(obj.diag(obj.l_ind,k-1),obj.diag(obj.c_ind,k));
```

```
157                         l = 1;
158                         temp_k=-1;
159                         while l <= L && k<n
160                             k = k + 1;
161                             if obj.sing_col(k)==0
162                                 obj.sing_col(k) = index_of_current_singularity;
163                             elseif temp_k==-1 && obj.sing_col(k) ~= ...
164                                     index_of_current_singularity
165                                 temp_k = k - 1;
166                             end
167                             obj.diag(obj.c_ind,k) = obj.diag(obj.l_ind,k-2)+...
168                                 1/(obj.diag(obj.c_ind,k-1)-obj.diag(obj.l_ind,k-1));
169                             l = l + 1;
170                         end
171                         obj.sing(obj.sing_col(k)).storeN(obj.diag(obj.l_ind,k-1));
172                         if temp_k > -1
173                             k = temp_k;
174                         end
175                         %%% END BLOCk 2
176                     end
177                 end
178                 %
179                 %
180                 k = k + 1;
181             end
182             %
183             if obj.debug
184                 obj.save_current_diag();
185             end
186         end
187
188         function save_current_diag(obj)
189             n = obj.lastrow;
190             k = 2;
191             while n>0 && k<=obj.col
192                 obj.eps(n,k) = obj.diag(obj.c_ind,k);
193                 n = n-1;
194                 k = k+1;
195             end
196         end
197
198         function al = get_ratio_for_STEA2(obj, two_k)
199             al = (obj.diag(obj.c_ind,two_k+2)-obj.diag(obj.l_ind,two_k))...
200                     /(obj.diag(obj.c_ind,two_k)-obj.diag(obj.l_ind,two_k));
201
202
203         end
204
205         function n_sing = get_number_of_singularity(obj)
206             n_sing = obj.n_sing;
207         end
208
209         function M = check_all_sing(obj)
210             if obj.debug
211                 M = false(size(obj.eps));
212                 for i=2:(obj.lastrow)
213                     for j=2:min(obj.lastrow-i+2,obj.col)
214                         if abs(obj.eps(i,j)-obj.eps(i-1,j))<...
215                                 10^obj.p*abs(obj.eps(i-1,j))
216                             M(i-1,j) =  true;
217                             M(i,j) = true;
218                         end
219                     end
220                 end
221             end
```

```
222         end
223
224         function print_matrix_eps(obj)
225             if obj.debug
226                 M = check_all_sing(obj);
227                 for i=1:obj.lastrow
228                     for j=2:obj.lastrow-i+2
229                         color = M(i,j) + 1; %if 1 then black elif 2 then red
230                         if isfinite(obj.eps(i,j))
231                             if obj.eps(i,j)<0
232                                 fprintf(color,'%.4e ', obj.eps(i,j));
233                             else
234                                 fprintf(color,' %.4e ', obj.eps(i,j));
235                             end
236                         else
237                             fprintf(color,'    %e     ',obj.eps(i,j));
238                         end
239                         fprintf(' ');
240                     end
241                     fprintf('\n');
242                 end
243             end
244         end
245
246         function M = return_eps(obj)
247             M = obj.eps;
248         end
249
250         function print_tree_eps(obj)
251             if obj.debug
252                 M = check_all_sing(obj);
253                 for r = 1:obj.lastrow
254                     %first row
255                     i = r; j=2;
256                     while i>0 && j<=obj.col && i+j<=obj.lastrow+2
257                         color = M(i,j) + 1; %if 1 then black elif 2 then red
258                         if isfinite(obj.eps(i,j))
259                             if obj.eps(i,j)<0
260                                 fprintf(color,'%.4e ', obj.eps(i,j));
261                             else
262                                 fprintf(color,' %.4e ', obj.eps(i,j));
263                             end
264                         else
265                             fprintf(color,'    %e     ',obj.eps(i,j));
266                         end
267                         i = i-1; j=j+2;
268                         fprintf('            ');
269                     end
270                     fprintf('\n');
271                     %second row
272                     i = r; j=3;
273                     fprintf('            ')
274                     while i>0 && j<=obj.col && i+j<=obj.lastrow+2
275                         color = M(i,j) + 1; %if 1 then black elif 2 then red
276                         if isfinite(obj.eps(i,j))
277                             if obj.eps(i,j)<0
278                                 fprintf(color,'%.4e ', obj.eps(i,j));
279                             else
280                                 fprintf(color,' %.4e ', obj.eps(i,j));
281                             end
282                         else
283                             fprintf(color,'    %e     ',obj.eps(i,j));
284                         end
285                         i = i-1; j=j+2;
286                         fprintf('            ');
```

```
287                     end
288                     fprintf('\n');
289                 end
290             end
291         end
292     end
293
294 end
```

```
1  %STEA.m
2  classdef STEA2 < handle
3
4      properties (Access = private)
5          scalar_eps    % Scalar Epsilon Algorithm associated
6          n             % length of vector
7          y             % dual paramether
8          k             % number of columns
9          n_vectors     % number of vectors enter
10         diag          % tensor where last two 'diagonals' \cap{eps}
11                       % are stored
12         c_ind         % index of row in diag matrix for the current diag
13         l_ind         % index of row in diag matrix for the last diag
14         debug
15     end
16
17     methods
18         function obj = STEA2(N,k,y,debug)
19             debug = true;
20             if nargin==3
21                 debug = false;
22             end
23             obj.scalar_eps = SEA(N,k,debug);
24             obj.debug = debug;
25             obj.y = y;
26             obj.k = k;
27             obj.n_vectors = 0;
28             obj.n = length(y);
29             obj.diag = zeros(2,obj.n,1+k/2);
30             obj.c_ind = 2; obj.l_ind = 1;
31         end
32
33         function insert_new_vector(obj,x)
34             obj.n_vectors = obj.n_vectors + 1;
35             obj.l_ind = obj.c_ind; obj.c_ind = 3 - obj.c_ind;
36             obj.diag(obj.c_ind,:,1) = x;
37             obj.scalar_eps.insert_new_S(dot(x,obj.y));
38             numb_iter = min(floor((obj.n_vectors+1)/2),1+obj.k/2)-1;
39             for t = 1:numb_iter
40                 al = obj.scalar_eps.get_ratio_for_STEA2(2*t);
41                 obj.diag(obj.c_ind,:,t+1) = (1-al)*obj.diag(obj.l_ind,:,t)...
42                     + al*obj.diag(obj.c_ind,:,t);
43             end
44         end
45
46         function print_tree_scalar_eps(obj)
47             obj.scalar_eps.print_tree_eps();
48         end
49
50         function x_extra = extrapolation(obj)
51             ind = min(floor((obj.n_vectors+1)/2),1+obj.k/2);
52             x_extra = obj.diag(obj.c_ind,:,ind)';
53         end
54
```

```
55          function n_sing = get_number_of_singularity(obj)
56              n_sing = obj.scalar_eps.get_number_of_singularity();
57          end
58      end
59
60  end
```

Bibliography

[1] A. Aitken, *On Bernoulli's numerical solution of algebraic equations*, P. Roy. Soc. Edinb. **46** (1926), 289-305.

[2] A. Alipanah, S. Esmaeili, *Numerical solution of the two-dimensional Fredholm integral equations using gaussian radial basis function*, J. Comp. and Appl. Math. **235** (2011), 5342-5347.

[3] D.G. Anderson, *Iterative procedures for nonlinear integral equations*, J. Assoc. Comp. Mah. **12** (1965), 547-560.

[4] P. Assari, M. Dehghan, *On the numerical solution of nonlinear integral equations on non-rectangular domains utilizing thin plate spline collocation method*, Proc. Indian Acad. Sci. (2019) 129:83.

[5] V. Berinde, *Iterative Approximation of Fixed Points*, Springer-Verlag Berlin Heidelberg, Baia Mare, 2007.

[6] A.H. Borzabadi and O.S. Fard, *A numerical scheme for a class of nonlinear Fredholm integral equations of the second kind* J. Comp. Appl. Math. **232** (2009), 449-454.

[7] C. Brezinski, *Généralisation de la transformation de Shanks, de la table de Padé et de l'ε-algorithme*, Calcolo **12** (1975), 317-360.

[8] C. Brezinski, M. Redivo-Zaglia, *EPSfun: a Matlab toolbox for the simplified topological ε-algorithm*, Netlib (2017). (http://www.netlib.org/numeralgo/), na44 package.

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

[10] C. Brezinski, M. Redivo-Zaglia, *Extrapolation methods for the numerical solution of non-linear Fredholm integral equations*, J. Int. Eq. Appl., Spring 2019.

[11] C. Brezinski, M. Redivo-Zaglia, *The simplified topological ε-algorithms for accelerating sequences in a vector space*, SIAM J. Sci. Comp. **36** (2014), A2227 - A2247.

[12] C. Brezinski , M. Redivo-Zaglia, *The simplified topological ε-algorithms: software and applications.*, Numer. Alg. **74(4)** (2017), 1237-1260.

[13] R. M. Brooks, K. Schmitt, *The contraction mapping principle and some applications*, Electr. J. of Diff. Eq. (2009), monograph 09.

[14] F. Cordellier, *Démonstration algébrique de l'extension de l'identité de Wynn aux tables de Padé non normales*, Padé Approximation and its Applications **765**, 36-60.

[15] J.P. Délahaye, B. Germain-Bonne, *Résultats négatifs en accélération de la convergence*, Numerische Mathematik **35(4)** (1980), 443-457.

[16] H. Guoqianga, W. Jiongb, *Extrapolation of Nyström solution for two dimensional nonlinear Fredholm integral equations*, J. Comp. Appl. Math. **134** (2001), 259-268.

[17] Y. Ma, J. Huang, Hu Li, *A novel numerical method of two-dimensional Fredholm integral equations of the second kind*, Mathematical Problems in Engineering (2015).

[18] W. Hackbusch, *Integral equations, Theory and Numerical Treatment*, Birkhäuser, Berlin, 1995.

[19] H. Hochstadt, *Integral equations*, John Wiley & Sons, New York, 1973.

[20] 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.

[21] A. Karapiperi, *Extrapolation methods and their applications in numerical analysis and applied mathematics*, PHD Thesis (2016).

[22] K. J. Kim, M. Song, *Symmetric quadrature formulas over a unit disk*, Korean J. Comp. e Appl. Math. **4** (1997), 179-192.

[23] K. Maleknejad, M. Karami *Numerical solution of non-linear Fredholm integral equations by using multiwavelet in Petrov-Galerkin method*, Appl. Math. Comput. **168** (2005), 102-110.

[24] D. O'Regan, M. Meehan, *Existence theory for nonlinear integral and integrodifferential equations*, Kluwer Academic Publishers, Dordrecht, 1998.

[25] A. Polyanin, A. Manzhirov, *Handbook of Integral Equations*, Chapman & Hall/CRC, 2008.

[26] S. B. Pomeranz, *Aitken's Δ^2 method extended*, Cogent Mathematics **4** (2017).

[27] D. Shanks, *Non linear transformations of divergent and slowly convergent sequences*, J. Math. Phys. **34(1-4)** (1955), 1-42.

[28] A. Sommariva, *Almost minimal rules on the square (Legendre weight)*, a several pointset suitable for cubature on disk. (`https://www.math.unipd.it/~alvise/sets.html`).

[29] P. Wynn, *On a device for computing the $e_m(S_n)$ transformation*, Math. Tables A. Comp. **10** (1956), 91-96.

[30] P. Wynn, *Singular rules for certain nonlinear algorithms*, BIT Numerical Mathematics **3** (1963), 175-195.

[31] P. Wynn, *Acceleration techniques for iterated vector and matrix problems*, Math. Comput., **16** (1962), 301-322.

[32] P.P. Zabreyko et al., *Integral equations: a reference text*, Noordhoff International Publishing, Leyden, 1975.