Università
degli Studi
di Padova

Sede Amministrativa: Università degli Studi di Padova
Dipartimento di Matematica

SCUOLA DI DOTTORATO DI RICERCA IN SCIENZE MATEMATICHE
INDIRIZZO: MATEMATICA COMPUTAZIONALE
CICLO: XXVII

# EXTRAPOLATION METHODS AND THEIR APPLICATIONS IN NUMERICAL ANALYSIS AND APPLIED MATHEMATICS

**Direttore della Scuola:** Ch.mo Prof. Pier Paolo Soravia
**Coordinatore d'indirizzo:** Ch.ma Prof.ssa Michela Redivo Zaglia
**Supervisore:** Ch.ma Prof.ssa Michela Redivo Zaglia

**Dottoranda:** Anna Karapiperi

## Riassunto

Questa tesi di dottorato tratta alcune applicazioni dei metodi di estrapolazione. Spesso in analisi numerica e nella matematica applicata si devono trattare successioni che convergono lentamente al loro limite. I metodi di estrapolazione possono essere utilizzati per accelerare la convergenza di una successione che converge lentamente o anche per sommare serie divergenti.

I primi due capitoli della tesi sono dedicati alle trasformazioni di successioni scalari. Viene ripreso il $\Delta^2$ di Aitken e vengono proposte tre nuove trasformazioni che lo generalizzano. Le proprietà di convergenza e di accelerazione di una delle trasformazioni sono discusse teoricamente e verificate sperimentalmente usando delle successioni divergenti e convergenti. La trasformazione di Shanks e l'$\varepsilon$-*algorithm* di Wynn sono accuratamente studiati; vengono richiamate le regole particolari proposte da Wynn per il trattamento delle singolarità isolate, ovvero quando due elementi consecutivi sono uguali o quasi uguali, ed anche le regole particolari, più generali, proposte da Cordellier, per il trattamento delle singolarità non isolate, ovvero quando più di due elementi sono uguali. Viene proposta una nuova generale implementazione delle regole particolari in modo da poter trattare tutti i casi possibili, ossia la presenza di singolarità causata da due o più elementi che sono uguali o quasi uguali.

Nella parte rimanente della tesi ci si concentra sull'estrapolazione vettoriale. Prima vengono brevemente descritti l'$\varepsilon$-*algorithm* vettoriale, l'$\varepsilon$-*algorithm* topologico e la sua versione semplificata, recentemente introdotta da Brezinski e Redivo-Zaglia. Successivamente, vengono presentate, con una notazione unificata le *Algebraic Reconstruction Techniques* (ART), le *Simultaneous Iterative Reconstruction Techniques* (SIRT) e altri metodi iterativi di regolarizzazione, che sono comunemente utilizzati per risolvere problemi inversi lineari. Infine, vengono illustrati i vantaggi ottenuti applicando l'estrapolazione ai precedenti metodi iterativi, utilizzati su problemi relativi alle immagini. In particolare, viene utilizzato il *simplified topological* $\varepsilon$-*algorithm* al fine di estrapolare una successione generata da metodi di tipo Landweber e Cimmino quando si risolvono problemi di ricostruzione e di restauro di immagini. I risultati numerici mostrano un buon comportamento dei metodi accelerati rispetto alle loro versioni non accelerate ed anche rispetto ad altri metodi.

# Abstract

This Ph.D. thesis discusses some applications of extrapolation methods. In numerical analysis and in applied mathematics one has often to deal with sequences which converge slowly to their limit. Extrapolation methods can be used to accelerate the convergence of a slow converging sequence or even to sum up divergent series.

The first two chapters of this thesis are devoted to scalar sequence transformations. We revisit Aitken's $\Delta^2$ process and we propose three new transformations which generalize it. The convergence and acceleration properties of one of our transformations are discussed theoretically and verified experimentally using diverging and converging sequences. Shanks transformation and Wynn's $\varepsilon$-algorithm are studied extensively; we remind the particular rules due to Wynn for treating isolated singularities, i.e. when two consecutive elements are equal or almost equal, and the more general particular rules proposed by Cordellier for treating non-isolated singularities, i.e. when more than two elements are equal. A new implementation of the generalized particular rule is given covering all the cases, namely singularities caused by two or more elements that are equal or almost equal.

In the remaining part of the thesis we focus on vector extrapolation. First we briefly describe the vector $\varepsilon$-algorithm, the topological $\varepsilon$-algorithm and the simplified topological $\varepsilon$-algorithm, which was recently introduced by Brezinski and Redivo-Zaglia. In the sequel, we present under a unified notation the Algebraic Reconstruction Techniques, the Simultaneous Iterative Reconstruction Techniques, and other iterative regularization methods, which are commonly used for solving linear inverse problems. Last, we study the gain of applying extrapolation on these methods in imaging problems. In particular, we use the simplified topological $\varepsilon$-algorithm in order to extrapolate a sequence generated by methods such as Landweber's and Cimmino's when solving image reconstruction and restoration problems. The numerical results illustrate the good performance of the accelerated methods compared to their unaccelerated versions and other methods.

## Acknowledgements

This thesis marks the completion of the doctoral studies conducted at the Department of Mathematics, University of Padova, under the supervision of Professor Michela Redivo Zaglia. My research was supported by Foundation Cariparo (Cassa di Risparmio di Padova e Rovigo) Grant PARO 117288/11, and partially supported by INdAM, GNCS (Gruppo Nazionale per il Calcolo Scientifico), and by the "ex-60%" funds of the University of Padova. At this point, I would like to thank all the people working in Torre Archimede for creating a nice environment in which I was happy to work.

This Ph.D. was a nice adventure which I would not have experienced if it was not for three people each of whom played a special role. First I would like to mention prof. Marilena Mitrouli who believed in me and encouraged me to continue my studies abroad. Meeting prof. Claude Brezinski, attending his talks and discussing with him was inspiring and motivated me to work in extrapolation methods. This became true with the help of prof. Michela Redivo Zaglia who gave me the opportunity to work on this project under her supervision. I would like to thank her for her guidance in my research and for teaching me how to stand in my own two feet. She also encouraged me to expand my knowledge and my professional horizons, by attending seminars and participating in conferences. I also thank prof. Claude Brezinski and prof. Michela Redivo Zaglia for trusting me their software in extrapolation methods and for being available to answer my questions on extrapolation methods and their applications.

Other people that helped me in my Ph.D. research by engaging in many stimulating and fruitful discussions, are prof. Paolo Novati, prof. James Nagy, prof. Michele Benzi, Dr. Maria Rosaria Russo. Special thanks Dr Silvia Gazzola, Dr Stefano Pozza and Dr Davide Buoso for the great collaboration.

Apart from the scientific gain, studying in Padova offered me the chance to meet a lot of wonderful people. My dear friends, thank you for your warm welcome, your company, your support, your friendship. Of course, I never forget my old friends from Greece. They are always present even when they are miles away.

Last, I would like to express my gratitude to my family. I really want to thank them for their continuous support. During all these years they have helped me in all possible ways. Their presence reminds me what really matters in life. This thesis is dedicated to my husband. He supported me from the first moment and he kept believing in me even when I was losing my faith in myself. Lefteris, you are my power!

# Contents

# Introduction

In numerical analysis and in applied mathematics, often we have to deal with slow converging sequences and series. As a result, we have to compute a big amount of terms in order to obtain a good approximation. This problem can be solved by using convergence acceleration methods which are based on the idea of extrapolation. In this thesis we focus on the acceleration that a sequence transformation can bring. More precisely, we use a sequence transformation in order to transform the initial sequence into another one, which, under some assumptions, converges faster to the same limit. In other words, if the transformation is able to accelerate the convergence of the initial sequence, then a smaller number of terms needs to be computed and this enables to estimate the limit faster and sometimes more accurately. We would like to stress that sequence transformations are also used for summing up divergent sequences and series.

It has been shown that a universal transformation able to accelerate any sequence cannot exist (see [30, 31]). For this reason, in practical situations it is better to construct new transformations depending on the specific class of sequences that we want to accelerate. Of course, if a class is too small, the transformation will be very specific and it will be able to accelerate only a limited number of sequences. On the other hand, numerical experiments have shown that such a specialization provides better acceleration properties of the transformation when applied to certain sequences.

Several works are available in the literature about sequence transformations (e.g. [14, 16, 18, 30, 60, 69, 73]) and how to use them in practical situations (e.g. [11, 54, 56, 67]). In particular, [61, 62, 63] study extensively vector extrapolation algorithms. See also [37] for a discussion on the efficiency of several numerical techniques in the evaluation of power series expansions for special functions.

In this thesis we study the most significant scalar sequence transformations. We also construct some new transformations for the acceleration of a special class of sequences.

Furthermore, we revise certain vector extrapolation methods. For one of them we test its performance when applied to well-known iterative regularization methods. This issue is related to one of the numerous applications of extrapolation methods, namely the solution of a system of equations. More precisely, when we use an iterative method for approximating the solution of a system, we obtain a sequence which we will attempt to accelerate by using a vector extrapolation algorithm.

The main contributions of this thesis are summarized in the following list:

- The introduction of a new algorithm for implementing properly the general particular rules of the $\varepsilon$-algorithm so that the algorithm becomes more efficient.

- We introduce some new scalar sequence transformations, which generalize Aitken's $\Delta^2$ process.

- We revise under a unifying framework all the Algebraic Reconstruction Techniques, the Simultaneous Iterative Reconstruction Techniques, and other known iterative regularization methods. We emphasise on Cimmino's method, presenting the original work of Cimmino in a more comprehensible way, introducing a general formula and a new variant.

- we apply a general extrapolation method, namely the simplified topological $\varepsilon$-algorithm, to various iterative methods and we explore the performance in imaging problems. We give some insight into the acceleration properties of some particular methods and we provide some informal and additional insight into the regularized properties of the proposed strategies. We study the choice of the vector $\mathbf{y}$ used in the definition of the simplified topological $\varepsilon$-algorithm and we propose some "good" choices considering a certain extrapolated method.

The contents of each chapter are presented below:

- *Chapter 1:* We revise some basic scalar extrapolation methods and algorithms, namely $\Delta^2$ process, Shanks transformation and $\varepsilon$-algorithm. The particular rules of Wynn are revised.

- *Chapter 2:* Cordellier's particular rules which extend those of Wynn are studied. The algorithm that implements the $\varepsilon$-algorithm with the general particular rules is extended, so that it can treat any kind of singularities.

- *Chapter 3:* We construct three transformations that aim to the acceleration of a certain class of sequences generalizing $\Delta^2$ process. One of the transformations is studied further theoretically and its convergence and acceleration properties are analyzed. Numerical experiments compare the performance of the proposed transformations, also with other known scalar sequence transformations and illustrate the good performance in the convergence acceleration of both diverging and converging sequences.

- *Chapter 4:* We briefly describe the vector $\varepsilon$-algorithm of Wynn, the topological $\varepsilon$-algorithms of Brezinski, and the simplified topological $\varepsilon$-algorithms recently proposed by Brezinski and Redivo-Zaglia.

- *Chapter 5:* We revise the Algebraic Reconstruction Techniques (ART) and the Simultaneous Iterative Reconstruction Techniques (SIRT). We study separately Cimmino's method, for which we propose a generalization and discuss several variants among which a new one. The projected SIRT methods and the semi-iterative methods are also recalled.

- *Chapter 6:* We use the simplified topological $\varepsilon$-algorithm in order to accelerate the convergence of several iterative regularization methods. Apart from the numerical tests which are focused on imaging problems, we provide an insight and algorithmic details about the simplified topological $\varepsilon$-algorithm (STEA) applied to Cimmino and Landweber methods. We also develop a theoretical study on the acceleration properties of a particular extrapolated method and we provide some informal and additional insight into the regularized properties of the proposed strategies. The choice of the vector $\mathbf{y}$ involved in the formulation of STEA is discussed.

The author would like to stress that the results presented in Chapter 3 are published in [23]. Also, the material presented in Chapter 6 is part of the work originally developed in the submitted paper [35].

**Notation**. Throughout this thesis, elements of the vector space $\mathbb{R}^N$ (or a generic topological space) are written in bold, while regular typeface corresponds to scalars. Unless otherwise stated, we denote the $i$th component of a vector $\mathbf{z} \in \mathbb{R}^N$ by $z_i$, $i =$

$1, \ldots, N$. All vectors are column vectors, $\mathbf{a}_i = A^T \mathbf{e}_i$ is the column vector formed by the $i$th row of $A$, $\mathbf{e}_i$ is the $i$th canonical basis vector of appropriate dimension, and $I$ is an identity matrix of appropriate size. The notation $\langle \mathbf{y}, \mathbf{x} \rangle = \mathbf{y}^T \mathbf{x}$ stands for the standard inner product on $\mathbb{R}^N$, and $\|\mathbf{x}\|$ denotes the Euclidean vector norm. The spectral radius (the largest positive eigenvalue) of a matrix $A$ is denoted by $\rho(A)$. $\mathcal{N}(A)$ stands for the null space of the matrix A, and $\mathcal{R}(A)$ is the range or column space of the matrix $A$.

# Chapter 1

# Scalar extrapolation methods and algorithms

In this chapter we give an insight into sequence transformations (or extrapolation methods), which are used in order to estimate more efficiently the limit of a convergent sequence or to sum up divergent sequences and series. First, we revise some well-known scalar extrapolation methods, namely $\Delta^2$ process and Shanks transformation. We extensively study the scalar $\varepsilon$-algorithm, one of the classical extrapolation algorithms used for the implementation of Shanks transformation. We focus on the particular rules given by Cordellier [28] who extended Wynn's particular rules [76] to the case of an arbitrary number of equal quantities in the $\varepsilon$-array. The chapter concludes with an appropriate Matlab code covering all the cases.

## 1.1 An introduction in sequence transformations

Let $(S_n)$ be a sequence of (real or complex) numbers which converges to $S$. We may use a sequence transformation $T$ and transform the initial sequence $(S_n)$ into another sequence $(T_n)$. An example of a scalar sequence transformation is Aitken's $\Delta^2$ process [3]

$$T_n = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n}, \quad n = 0, 1, \ldots \tag{1.1}$$

In order to present some practical interest, the new sequence $(T_n)$ should

✓ converge,

✓ have the same limit as $(S_n)$,

✓ converge faster than $(S_n)$, that is $\lim_{n\to\infty} \dfrac{T_n - S}{S_n - S} = 0$.

If the last relation is satisfied, we say that the transformation $T$ accelerates the convergence of the sequence $(S_n)$.

The three aforementioned properties usually do not hold for all the convergent sequences $(S_n)$. Especially the last property is the most difficult to be satisfied for all the sequences. In fact, it has been proved that a transformation able to accelerate any sequence does not exist (see [30, 31]). Instead, every sequence transformation is only able to accelerate the convergence of certain classes of sequences. For example, Aitken's process accelerates the convergence of all the sequences for which $\exists\ \lambda \in [-1, +1[$ such that

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

However, for this transformation the first two properties are not satisfied for all convergent sequences. In [18, Section 2.3] one can find examples of convergent sequences $(S_n)$ for which the sequence $(T_n)$ obtained by Aitken's process has two accumulation points. On the other side, it is true that if such a $(T_n)$ converges, then its limit is the same as the limit of the sequence $(S_n)$ (see [68]).

In conclusion, we may say that in practical situations it is preferable to develop new algorithms depending on the class of sequences of interest. Apparently, if this class is too small, such transformation will be very specific and hence it will be useful only in particular cases; on the other hand, such specialization typically provides better acceleration properties. For this reason, in the study of a sequence transformation the first question to be asked and solved (before those of convergence and acceleration) is which is the kernel. The *kernel* $\mathcal{K}_T$ of a transformation $T : (S_n) \longmapsto (T_n)$ is defined as the set of all the sequences $(S_n)$ which are transformed by $T$ into a constant sequence, which is usually the limit (if it exists) of the sequence $S_n$,

$$\text{i.e. } \exists S \text{ such that } T_n = S,\ \forall n \geq N \text{ (for some } N > 0).$$

For instance, it has been proved that the kernel of $\Delta^2$ process consists of all and only the sequences of the form

$$S_n = S + a\lambda^n, \quad n = 0, 1, \ldots \tag{1.2}$$

where $a$ and $\lambda$ are scalars such that $a \neq 0$ and $\lambda \neq 0, 1$. In general, sufficiency is difficult to prove. We refer to [18, 20] for a detailed discussion on this transformation.

The equation (1.2) is the *explicit form* of the kernel since it gives explicitly the form of the sequences belonging to the kernel of Aitken's process. An equivalent expression is the so-called *implicit form* of the kernel, which for $\Delta^2$ process is described by the next relation

$$S_{n+1} - S = \lambda(S_n - S) \tag{1.3}$$

The solution of the above difference equation is given by (1.2).

If the sequence to be accelerated belongs to the kernel of the transformation we use, then, by construction, $\forall n \geq N,\ T_n = S$. Although it has not yet been proved, numerical experiments have always confirmed that the "closer" a sequence is to the kernel, the faster the transformed sequence will converge (to the same limit).

Note that usually $S$ is the limit of the sequence $(S_n)$ but this is not always the case. For example, in Aitken's process, $S$ is the limit of $(S_n)$ if $|\lambda| < 1$; otherwise, for $(S_n)$ diverging, $S$ is called the *antilimit* of the sequence. If $|\lambda| = 1$, then $(S_n)$ has no limit at all or it only takes a finite number of distinct values and $S$ is their arithmetical mean.

Now that we have explained everything about the kernel of a transformation, we may answer to the question *What an extrapolation method is.* A sequence transformation $T : (S_n) \longmapsto (T_n)$ is said to be an extrapolation method if it is such that

$$\forall n \geq N,\ T_n = S \quad \text{if and only if} \quad (S_n) \in \mathcal{K}_T.$$

In other words, any sequence transformation is an extrapolation method.

Next we will explain how a transformation $T$ is built from its kernel. In particular, we will study the simple case of Aitken's $\Delta^2$ process.

## 1.2 Aitken's $\Delta^2$ process

The most simple sequence transformation that we will treat in this thesis is $\Delta^2$ process (1.1). Let us see how this transformation can be derived from the kernel of $\Delta^2$ process. We consider (1.3) and we write it under the following more general form

$$a_0(S_n - S) + a_1(S_{n+1} - S) = 0, \quad n = 0, 1, \ldots \tag{1.4}$$

with $a_0 \, a_1 \neq 0$ and $a_0 + a_1 \neq 0$. In order to compute $S$ we need to know the value of $a_0$, $a_1$. We have, for all $n$,

$$a_0 \Delta S_n + a_1 \Delta S_{n+1} = 0, \tag{1.5}$$

where $\Delta$ is defined by $\Delta S_n = S_{n+1} - S_n$ and $\Delta^{k+1} S_n = \Delta^k S_{n+1} - \Delta^k S_n$, $n = 0, 1, \ldots$. Adding to the equation (1.5) the condition $a_0 + a_1 = 1$ we obtain a system in the unknowns $a_i$. Once we know the coefficients $a_i$, we can compute $S$ as follows

$$S = a_0 S_n + a_1 S_{n+1}. \tag{1.6}$$

Writing (1.6) for the indices $n$, $n+1$, we obtain the system

$$\begin{cases} S &= a_0 S_n + a_1 S_{n+1} \\ S &= a_0 S_{n+1} + a_1 S_{n+2} \end{cases} \tag{1.7}$$

If $S_n$ does not belong to the kernel (1.4), we can still develop the above procedure. The only difference is that now both the coefficients $a_i$ and the linear combination $a_0 S_n + a_1 S_{n+1}$ will depend on $n$ and $k$.

Since $T_n = S$, we can write

$$\begin{cases} T_n &= a_0 S_n &+ a_1 S_{n+1} \\ T_n &= a_0 S_{n+1} &+ a_1 S_{n+2} \end{cases} \tag{1.8}$$

We add and subtract $S_n$ to the first equation and $S_{n+1}$ to the second one. Being $a_0 + a_1 = 1$, we set $b = a_0 - 1$ and we have

$$\begin{cases} S_n &= T_n + b \Delta S_n \\ S_{n+1} &= T_n + b \Delta S_{n+1} \end{cases} \tag{1.9}$$

Using Cramer's rule for the solution of a system of linear equations, we write $T_n$ as the following ratio of determinants

$$T_n = \frac{\begin{vmatrix} S_n & S_{n+1} \\ \Delta S_n & \Delta S_{n+1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ \Delta S_n & \Delta S_{n+1} \end{vmatrix}}$$

The computation of the determinants gives

$$T_n = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2 S_{n+1} + S_n}, \quad n = 0, 1, \ldots \tag{1.10}$$

Note that the denominator is $\Delta^2 S_n = \Delta S_{n+1} - \Delta S_n$ and this explains the name $\Delta^2$ *process*. Alexander Craig Aitken (1895-1967) used this method in [3] (1926), so it is also named after Aitken. However, the $\Delta^2$ process was actually discovered by the Japanese mathematician Takakazu Seki (?-1708) before 1680. The same method was obtained by Hans von Naegelsbach (1838-?) in 1876 and by James Clerk Maxwell (1831-1879) in 1873 but neither of them used it for the purpose of acceleration (see e.g. [15, 55] and the references therein).

Formula (1.10) is highly numerically unstable since, if $S_n$, $S_{n+1}$ and $S_{n+2}$ are almost equal when $n$ tends to infinity, cancellation errors arise both in the numerator and in the denominator and $T_n$ is badly computed [18]. For that reason, in practice, we use one of the following equivalent formulas

$$T_n = S_n - \frac{(\Delta S_n)^2}{\Delta^2 S_n} = S_{n+1} - \frac{\Delta S_n \Delta S_{n+1}}{\Delta^2 S_n} = S_{n+2} - \frac{(\Delta S_{n+1})^2}{\Delta^2 S_n}, \quad n = 0, 1, \ldots \tag{1.11}$$

Of course, cancellation errors again arise but in the correcting term to $S_n$, $S_{n+1}$, $S_{n+2}$ respectively, thus formulas (1.11) are much more stable than (1.10). Formulas (1.11) can be considered particular rules for $\Delta^2$ process which we use in order to avoid propagation of rounding errors due to the computer's arithmetic.

## 1.3 Shanks transformation

This is certainly one of the most familiar scalar extrapolation methods. It was found by Shanks [59] as a generalization of Aitken's $\Delta^2$ process. Its kernel is the set of sequences satisfying the homogeneous linear difference equation of order $k$

$$a_0 (S_n - S) + a_1 (S_{n+1} - S) + \cdots + a_k (S_{n+k} - S) = 0, \quad n = 0, 1, \ldots \tag{1.12}$$

where the coefficients $a_i$ are arbitrary constants independent of $n$ such that $a_0 a_k \neq 0$, $a_0 + \cdots + a_k \neq 0$. Note that if we write (1.12) for $k = 1$ we obtain (1.4) that is Aitken's $\Delta^2$ process.

We repeat the steps we followed for Aitken's process, assuming that $a_0 + \cdots + a_k = 1$, we obtain, a formula analogous to (1.6)

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

If $S_n$ does not belong to the kernel (1.12), we can still develop the same procedure, but now both the coefficients $a_i$ and the linear combination $a_0 S_n + \cdots + a_k S_{n+k}$ will depend on $n$ and $k$. For remembering this dependence we add to $a_i$ the superscript $(n,k)$ and we write

$$e_k(S_n) = a_0^{(n,k)} S_n + \cdots + a_k^{(n,k)} S_{n+k}, \quad k, n = 0, 1, \ldots \tag{1.13}$$

The coefficients $a_i^{(n,k)}$, $i = 1, \ldots, k$, are solution of the system

$$\left\{ \begin{array}{ccccccc} a_0^{(n,k)} & + & \cdots & + & a_k^{(n,k)} & = & 1 \\ a_0^{(n,k)} \Delta \mathbf{S}_n & + & \cdots & + & a_k^{(n,k)} \Delta \mathbf{S}_{n+k} & = & 0 \\ \vdots & & \vdots & & & & \vdots \\ a_0^{(n,k)} \Delta \mathbf{S}_{n+k-1} & + & \cdots & + & a_k^{(n,k)} \Delta \mathbf{S}_{n+2k-1} & = & 0 \end{array} \right. . \tag{1.14}$$

Cramer's rule gives

$$e_k(S_n) \quad = \quad \frac{\begin{vmatrix} S_n & S_{n+1} & \cdots & S_{n+k} \\ \Delta S_n & \Delta S_{n+1} & \cdots & \Delta S_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta S_{n+k-1} & \Delta S_{n+k} & \cdots & \Delta S_{n+2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \Delta S_n & \Delta S_{n+1} & \cdots & \Delta S_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta S_{n+k-1} & \Delta S_{n+k} & \cdots & \Delta S_{n+2k-1} \end{vmatrix}} \quad k, n = 0, 1, \ldots \tag{1.15}$$

By construction, we have the following result

**Theorem 1.1.** ([18]) *For all $n$, $e_k(S_n) = S$ if and only if $\exists a_0, a_1, \ldots, a_k$, with $a_0\, a_k \neq 0$ and $a_0 + \cdots + a_k \neq 0$, such that, for all $n$,*

$$a_0(S_n - S) + \cdots + a_k(S_{n+k} - S) = 0. \tag{1.16}$$

*In other words, for $k$ fixed $(S_n) \longmapsto (e_k(S_n))_n$ if and only if $(S_n)$ belongs to the kernel of Shanks transformation.*

It is clear that the determinants in (1.15) cannot be computed easily as in Aitken's process, since for a large $k$ such a computation would be prohibitive for the time required but mostly from a numerical point of view. So what we do in this case is to compute $e_k(S_n)$ recursively via an *extrapolation algorithm*. In particular, for the implementation of Shanks transformation, we use the well-known $\varepsilon$-algorithm of Wynn.

## 1.4 The scalar $\varepsilon$-algorithm

One of the most important scalar extrapolation algorithms is $\varepsilon$-algorithm. Proposed by Wynn [75] for implementing Shanks transformation, the scalar $\varepsilon$-algorithm computes the numbers $\varepsilon_k^{(n)}$ by the following rules

$$\begin{cases} \varepsilon_{-1}^{(n)} = 0, \quad \varepsilon_0^{(n)} = S_n, & n = 0, 1, ... \\ \varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + (\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)})^{-1}, & k, n = 0, 1, ... \end{cases} \tag{1.17}$$

**Definition 1.1.** *We will refer to the relationship*

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + (\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)})^{-1} \tag{1.18}$$

*as the normal rule of the $\varepsilon$-algorithm.*

Its connection with Shanks transformation is described by the following relationship

$$\varepsilon_{2k}^{(n)} = e_k(S_n), \quad k, n = 0, 1, \ldots$$

The numbers $\varepsilon_{2k+1}^{(n)}$ are intermediate results and they are equal to $1/e_k(\Delta S_n)$. The numbers $\varepsilon_k^{(n)}$ are displayed in a double entry table, called the $\varepsilon$*-array*, in such a way that the index $k$ denotes a column and the superscript $n$ a descending diagonal. Notice that the sum of $k$ and $n$ is constant among an ascending diagonal and reveal the number of the diagonal (see Table 1.1).

Starting from the first two columns, the rule of the $\varepsilon$-algorithm allows to proceed in the $\varepsilon$-array from left to right and from top to bottom.

By eliminating the columns with the same parity, we obtain the cross rule due to Wynn (or Wynn's identity)

$$\left(\varepsilon_{k+2}^{(n)} - \varepsilon_k^{(n+1)}\right)^{-1} + \left(\varepsilon_{k-2}^{(n+2)} - \varepsilon_k^{(n+1)}\right)^{-1} = \left(\varepsilon_k^{(n+2)} - \varepsilon_k^{(n+1)}\right)^{-1} + \left(\varepsilon_k^{(n)} - \varepsilon_k^{(n+1)}\right)^{-1}, \ k, n \geq 0. \tag{1.19}$$

Of course, this rule is mathematically equivalent to the normal rule of the $\varepsilon$-algorithm. We recall that in the application of the $\varepsilon$-algorithm the only useful columns are the even, whilst the numbers $\varepsilon_{2k+1}^{(n)}$ are merely intermediate quantities. We can use Wynn's identity in order to compute only the even columns, adding the initial conditions

$$\begin{cases} \varepsilon_{-2}^{(n)} = \infty, & n \geq 2, \\ \varepsilon_0^{(n)} = S_n, & n \geq 0. \end{cases}$$

$$\begin{array}{cccccc}
\varepsilon_{-1}^{(0)} = 0 & & & & & \\
 & \varepsilon_0^{(0)} = S_0 & & & & \\
\varepsilon_{-1}^{(1)} = 0 & & \varepsilon_1^{(0)} & & & \\
 & \varepsilon_0^{(1)} = S_1 & & \varepsilon_2^{(0)} & & \\
\varepsilon_{-1}^{(2)} = 0 & & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} & \\
 & \varepsilon_0^{(2)} = S_2 & & \varepsilon_2^{(1)} & & \ddots \\
\varepsilon_{-1}^{(3)} = 0 & & \varepsilon_1^{(2)} & & \varepsilon_3^{(1)} & \\
\vdots & \varepsilon_0^{(3)} = S_3 & & \varepsilon_2^{(2)} & & \ddots \\
\vdots & \vdots & \varepsilon_1^{(3)} & & \varepsilon_3^{(2)} & \\
\vdots & \vdots & \vdots & \varepsilon_2^{(3)} & & \ddots \\
\vdots & \vdots & \vdots & \vdots & \varepsilon_3^{(3)} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Table 1.1: The $\varepsilon$-array

Let us now come back to the normal rule of the $\varepsilon$-algorithm . We saw that the numbers $\varepsilon_{2k}^{(n)}$ are approximations of the limit $S$ of the sequence $(S_n)$. If the algorithm works well, then for some $k$ and $n$, $\varepsilon_{2k}^{(n)}$ and $\varepsilon_{2k}^{(n+1)}$ will be almost equal as they will be both good approximations of $S$. Therefore in the computation of $\varepsilon_{2k+1}^{(n)}$, an important cancellation error will occur in the difference $\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}$. Thus $\varepsilon_{2k+1}^{(n)}$ will be large and badly computed. Similarly, if $\varepsilon_{2k}^{(n+2)}$ is close to $S$, then $\varepsilon_{2k+1}^{(n+1)}$ will be large and badly computed, too. Therefore, when we try to compute $\varepsilon_{2k+2}^{(n)}$, the difference of two large and badly computed quantities appearing in the denominator will cause numerical instability in the algorithm. In order to avoid such kind of computational difficulties we use the particular rules of the $\varepsilon$-algorithm.

### 1.4.1 Particular rules

We already saw that the normal rule of the $\varepsilon$-algorithm (1.18) can suffer from a very serious drawback: cancellation errors due to the computer's arithmetic which result in a breakdown or a near-breakdown. This happens when two or more consecutive elements of the same column of the $\varepsilon$-array are equal or almost equal. In this case we say that we have a singularity. When the elements involved in the singularity are only two, then we call it an isolated singularity; otherwise we have a non-isolated singularity.

Let us first focus on the simple case of an isolated singularity. We assume that $\varepsilon_{k-1}^{(n+1)} = \varepsilon_{k-1}^{(n+2)} = \alpha$. Then the state of affairs is as shown in Table 1.2. The singularity arising in the column $k-1$, influences further values of the rhombus; $\varepsilon_{k}^{(n+1)}$ becomes formally $\infty$, $\varepsilon_{k+1}^{(n)}$ and $\varepsilon_{k+1}^{(n+1)}$ are equal to $\alpha$ and $\varepsilon_{k+2}^{(n)}$ is indeterminate.

| | | | | |
|---|---|---|---|---|
| | | $\varepsilon_{k}^{(n)}$ | | |
| | $(\alpha)\varepsilon_{k-1}^{(n+1)}$ | | $(\alpha)\varepsilon_{k+1}^{(n)}$ | |
| $\varepsilon_{k-2}^{(n+2)}$ | | $(\infty)\varepsilon_{k}^{(n+1)}$ | | $(*)\varepsilon_{k+2}^{(n)}$ |
| | $(\alpha)\varepsilon_{k-1}^{(n+2)}$ | | $(\alpha)\varepsilon_{k+1}^{(n+1)}$ | |
| | | $\varepsilon_{k}^{(n+2)}$ | | |

Table 1.2: An isolated singularity.

In [76] Wynn describes how to jump over such a singularity. When it comes to compute $\varepsilon_{k+2}^{(n)}$, instead of using the normal rule (1.18), he makes use of the particular rule

$$\varepsilon_{k+2}^{(n)} = a\left(1 + a/\varepsilon_{k}^{(n+1)}\right)^{-1},$$

where

$$a = \varepsilon_{k}^{(n+2)}\left(1 - \varepsilon_{k}^{(n+2)}/\varepsilon_{k}^{(n+1)}\right)^{-1} + \varepsilon_{k}^{(n)}\left(1 - \varepsilon_{k}^{(n)}/\varepsilon_{k}^{(n+1)}\right)^{-1} - \varepsilon_{k-2}^{(n+2)}\left(1 - \varepsilon_{k-2}^{(n+2)}/\varepsilon_{k}^{(n+1)}\right)^{-1} \tag{1.20}$$

If $\varepsilon_{k-1}^{(n+1)}$ and $\varepsilon_{k-1}^{(n+2)}$ are almost equal, then $\varepsilon_{k}^{(n+1)}$ is large and ill-determined, but $\varepsilon_{k+1}^{(n)}$ and $\varepsilon_{k+1}^{(n+1)}$ computed using the normal rule (1.18) are quite well determined, and so is $\varepsilon_{k+2}^{(n)}$ derived from (1.20). When $\varepsilon_{k-1}^{(n+1)}$ and $\varepsilon_{k-1}^{(n+2)}$ are exactly equal, the singular rule for the $\varepsilon$-algorithm reduces to

$$\varepsilon_{k+2}^{(n)} = \varepsilon_{k}^{(n+2)} + \varepsilon_{k}^{(n)} - \varepsilon_{k-2}^{(n+2)}. \tag{1.21}$$

In order to facilitate the discussion, we denote the elements of Table 1.2 by their cardinal position in the array ($C$ stands for center), as shown in Table 1.3.

With this notation, (1.19) can be written as

$$(E-C)^{-1} + (W-C)^{-1} = (S-C)^{-1} + (N-C)^{-1} \tag{1.22}$$

or, equivalently,

$$E = C + \left[(S-C)^{-1} + (N-C)^{-1} - (W-C)^{-1}\right]^{-1}. \tag{1.23}$$

| | | | | |
|---|---|---|---|---|
| | | $N$ | | |
| | $NW$ | | $NE$ | |
| $W$ | | $C$ | | $E$ |
| | $SW$ | | $SE$ | |
| | | $S$ | | |

Table 1.3

Using the normal rule of the $\varepsilon$-algorithm (1.18) we have

$$\begin{array}{rcrcl} C & = & W & + & 1/(SW-NW), \\ NE & = & NW & + & 1/(C-N), \\ SE & = & SW & + & 1/(S-C), \\ E & = & C & + & 1/(SE-NE). \end{array}$$

We observe that if $N=C$, then $NE$ is infinity, and if $S=C$, then $SE$ is infinity. Being $NE$ and $SE$ both infinity, $E$ is undefined and the computations have to be stopped. There is a breakdown in the algorithm. The same is true if $NW=SW$ since then $C$ is infinity. If $N \neq C$ , then $NE = NW = SW$. If $S \neq C$, then $SE = SW = NW = NE$ and $E$ is undefined. If $N$ is different from $C$ but very close to it, then a cancellation error arises in the computation of $NE$ which will be large and badly computed. The same is true for $SE$ if $S$ is different from $C$ but close to it. If $NW$ is different from $SW$ but close to it, $C$ will be large and badly computed. Thus $NE$ and $SE$ will be almost equal and $E$ will be different of two large and badly computed numbers. There is a near-breakdown in the algorithm.

With the new terminology, the particular rule (1.20) can be written as follows

$$\begin{gathered} E = r(1+r/C)^{-1}, \\ \text{where} \quad r = S(1-S/C)^{-1} + N(1-N/C)^{-1} - W(1-W/C)^{-1}. \end{gathered} \tag{1.24}$$

This rule was shown to be more stable than the rule given above for computing $E$. If $C$ is infinity, it reduces to

$$E = S+N-W \tag{1.25}$$

thus allowing to compute E by jumping over the singularity (or the breakdown). We recall that this rule is valid when there is only one isolated singularity that is when $N$ and $S$ are not infinity, or, equivalently, when only two adjacent quantities in a column ($NW$ and $SW$ in our example) are equal or almost equal.

# Chapter 2

# An extension of Wynn's particular rules

In the previous chapter we reminded the particular rules proposed by Wynn for treating isolated singularities. Cordellier [28] extended these rules for the case of an arbitrary number of equal quantities in the $\varepsilon$-algorithm. In this chapter we complete these rules discussing also the case of non-isolated singularities caused by almost equal values and we propose an algorithm for the implementation of $\varepsilon$-algorithm with the generalized particular rule.

## 2.1 Cordellier's particular rules

We recall that when three or more consecutive elements of the same column in the $\varepsilon$-array are equal or almost equal, we have a non-isolated singularity. An example of a singularity of size $m+1$, that is a singularity caused by $m+1$ (exactly or almost) equal values, is shown in Table 2.1. The quantities $N_i$, $S_i$ $(i = 0, \ldots, m+1)$, $W_i$, $E_i$ $(i = 1, \ldots, m)$, $\alpha$ are distinct from $\infty$; the Greek letter $\omega$ stands for an undefined number. The area affected by the singularity lies inside the square block, which we will call singular square block. In this case the number of (exactly or almost) equal values is $m+1$, thus we say that we have a singular square block of size $m+1$. When $m=1$ then we have a singularity of size 2 (isolated singularity) which was discussed in Section 1.4.1.

The quantities $N_i$, $S_i$, $W_i$, $E_i$ $(i = 1, \ldots, m)$ are related by the general rule (1.22) proposed by Cordellier (see [28, Proposition 9]),

$$(E_i - C)^{-1} + (W_i - C)^{-1} = (S_i - C)^{-1} + (N_i - C)^{-1}, \quad i = 1, \ldots, m. \tag{2.1}$$

For the computation of $E_i$'s we make use of the generalized particular rule described below

$$\begin{aligned} E_i &= r_i(1 + r_i/C)^{-1}, \quad i = 1, \ldots, m, \\ \text{where} \quad r_i &= S_i(1 - S_i/C)^{-1} + N_i(1 - N_i/C)^{-1} - W_i(1 - W_i/C)^{-1}. \end{aligned} \tag{2.2}$$

This rule generalizes Wynn's particular rule (1.24). When $C = \infty$, (2.2) reduces to

$$E_i = S_i + N_i - W_i, \quad i = 1, \ldots, m, \tag{2.3}$$

[28, Proposition 8], which is an extension of (1.25).

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

Table 2.1: Part of an $\varepsilon$-array with a singularity of size $m + 1$.

## 2.2 A new algorithm for general particular rules

For the implementation of his particular rules, Cordellier has proposed an algorithm (see [29, Annexe4, pp. 41-42] for the code in Pascal). This algorithm has a serious limitation: it only treats singularities caused by exactly equal values. This means that when almost equal values appear in the $\varepsilon$-array, no particular rule apply we face the well-known problem that the use of the normal rule may bring (i.e. cancellation errors due to the computer's arithmetic) and we obtain bad numerical results. Furthermore, Cordellier's algorithm sets all the elements inside a singular square block equal to $\alpha$. In this way, the more general cross rule (2.2), which uses the value $C$, is excluded. Of course, for Cordellier this was not a problem, since his algorithm was designed to use only formula (2.3), thus the inner values of the singular square block do not affect the values outside.

In this section we introduce a new algorithm which overcomes the aforementioned difficulties. In particular, the new algorithm treats as a singularity the case of two or more almost equal values in the $\varepsilon$-array. This is implemented via the control described below.

*Control for singularity.* If, for some fixed $p$,

$$\frac{|\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}|}{|\varepsilon_k^{(n)}|} < 10^{-p}, \quad k, n = 0, 1, \ldots \tag{2.4}$$

we have a singularity.

Right after the first control, another one is performed in order to check if the elements $\varepsilon_k^{(n)}$, $\varepsilon_k^{(n+1)}$ have exactly or almost equal values in order to use the appropriate particular rule for the computation of the $E_i$'s. But first we need to know where the $E_i$'s are located in the $\varepsilon$-array. For this reason the algorithm counts the number of equal consecutive elements. In practice, this means that every time that we meet a singularity, i.e. the elements $\varepsilon_k^{(n)}$ and $\varepsilon_k^{(n+1)}$ satisfy the condition (2.4), we check if $\varepsilon_k^{(n-1)}$ and $\varepsilon_k^{(n)}$ satisfy the same condition. If $\varepsilon_k^{(n)}$ and $\varepsilon_k^{(n+1)}$ are the first consecutive elements of their column that satisfy the above condition, then a new singular square block opens with the entry of the element $\varepsilon_k^{(n+1)}$; otherwise, the new element $\varepsilon_k^{(n+1)}$ contributes to a non-isolated singularity and increases by 1 the size of an old (still open) singular square block. As soon as we find an element $\varepsilon_k^{(n+m+1)}$ which, together with $\varepsilon_k^{(n+m)}$, do not satisfy the condition (2.4), we know that we have just left behind a singular square block of size $m+1$. Then the elements $E_i$, $i = 1, \ldots, m$, are located $2m+1$ columns on the right with respect to the column where the singularity occurs (see Table 2.1).

**Remark 2.1.** *In our implementation we use the normal rule* (1.18) *in order to compute the values inside a singular square block. However, when exactly equal values cause a non-isolated singularity, we have to impose the value $\infty$ to the places where $C$ appears* *in Table 2.1, otherwise once the first undefined value appears (due to the operation* $\infty - \infty$ *in the denominator of* (1.18)*), this will be propagated in the rest of the* $\varepsilon$*-array.*

Now, we can compute the values of $E_i$'s. If the singularity is caused by exactly equal values, then we use formula (2.3). In the case of almost equal values, the theory says that we use formula (2.2). However, note that in the case of almost equal values, $\alpha$ is not the same everywhere in the singular square block, thus $C$ also varies slightly. In our implementation, we used (2.2) with a different $C$, denote it $C_i$, for every $E_i$. In Table 2.2 we can see which quantity $C_i$ is used in the computation of $E_i$ and where the quantities $W_i, S_i, N_i, E_i, C_i$ are placed in the neighborhood of a singular square block. The arrows show the direction in which the lower index $i$ grows.

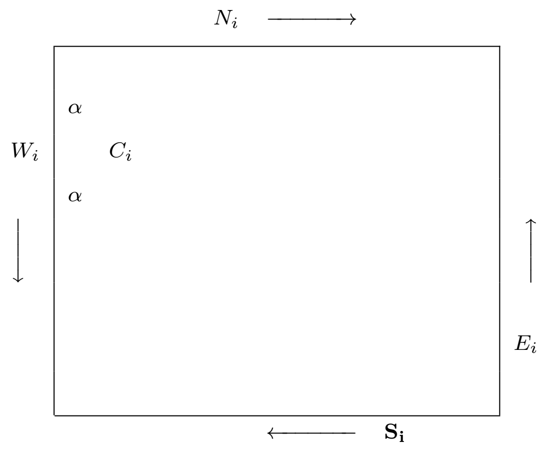


Table 2.2: The quantities $E_i$, $W_i$, $N_i$, $S_i$, $C_i$ related via formula (2.5).

Assuming that the singularity occurs at the column $k$ in the $\varepsilon$-array, e.g. $\varepsilon_k^{(n+i-1)} \simeq \varepsilon_k^{(n+i)} \simeq \alpha$, then the quantity $E_i$ ($i = 1, \ldots, m$, given by the size of the singular square block is $m+1$) which corresponds to the element $\varepsilon_{k+2(m+1)-1}^{(n-i)}$ is computed by the formula

$$\begin{aligned} E_i &= r_i(1 + r_i/C_i)^{-1}, \quad i = 1, \ldots, m, \\ \text{where} \quad r_i &= S_i(1 - S_i/C_i)^{-1} + N_i(1 - N_i/C_i)^{-1} - W_i(1 - W_i/C_i)^{-1}, \end{aligned} \tag{2.5}$$

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

### 2.2.1 Numerical experiments

In this section we test the new algorithm on four sequences; the first one has two isolated singularities and so Wynn's particular rules can be used, while in the second sequence several non-isolated singularities caused by exactly equal values are present and thus Cordellier's algorithm can successfully treat this case. Of course the new algorithm, containing as particular cases the algorithms of Wynn and Cordellier, can also be used for both sequences and it will give the same results. However, the last two examples we present contain non-isolated singularities caused by almost equal values. Therefore they can be treated only by the new algorithm.
**Notation**. Throughout this section we will use the notation `EPSALGO` for the standard $\varepsilon$-algorithm (which applies always the normal rule), `EPSALGOW` if Wynn's particular rules are used, `EPSALGOC` will stand for Cordellier's algorithm, while the new algorithm implementing the generalized particular rules will be denoted by `EPSALGOG`.

**Example 1.** [18, p.38] We consider the sequence given by

$$\begin{aligned} &S_0 = 1.59999999,\ S_1 = 1.2,\ S_2 = 1, \\ &S_n = \frac{S_{n-1}}{2} + \frac{S_{n-2}}{4} + \frac{S_{n-3}}{8} \text{ for n} \geq 3. \end{aligned} \tag{2.6}$$

According to Theorem 1.1, $\forall n$, $\varepsilon_6^{(n)} = 0$, since $e_3(S_n) = \varepsilon_6^{(n)}$. Let us see if the numerical results agree. We apply `EPSALGOW` using Wynn's particular rules whenever two successive quantities have 8 common digits (i.e. $\varepsilon_k^{(n)}$ and $\varepsilon_k^{(n+1)}$ satisfy (2.4) with $p = 8$). The resulting array is shown in Table 2.4. The two singular square blocks are marked and the values computed by the particular rule are written in bold. Concerning the values $\varepsilon_6^{(n)}$, $n = 0, 1, \ldots 5$, (values coloured in red), we observe that `EPSALGOW` algorithm gives a good approximation. In Table 2.5 we compare these results with those obtained by the $\varepsilon$-algorithm without using the particular rules. The comparison shows clearly the gain from using the particular rules. In Table 2.3 one can see the whole $\varepsilon$-array resulting from `EPSALGO` algorithm.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.6000e+00 | | | | | | | | | | | |
| | -2.5000e+00 | | | | | | | | | | |
| 1.2000e+00 | | 8.0000e-01 | | | | | | | | | |
| | -5.0000e+00 | | -2.1875e-07 | | | | | | | | |
| 1.0000e+00 | | 1.0000e+00 | | 4.5714e+06 | | | | | | | |
| | -8.0000e+08 | | 0 | | -2.1875e-07 | | | | | | |
| 1.0000e+00 | | 1.0000e+00 | | 9.2000e-01 | | 8.5200e-01 | | | | | |
| | -1.0000e+01 | | -1.2500e+01 | | -1.4706e+01 | | -1.5432e+01 | | | | |
| 9.0000e-01 | | 6.0000e-01 | | 4.6667e-01 | | -5.2500e-01 | | 1.5000e+00 | | | |
| | -1.3333e+01 | | -2.0000e+01 | | -1.5714e+01 | | -1.4938e+01 | | -1.5765e+01 | | |
| 8.2500e-01 | | 4.5000e-01 | | 7.0000e-01 | | 7.6364e-01 | | 2.9104e-01 | | 1.5449e-01 | |
| | -1.6000e+01 | | -1.6000e+01 | | -4.6229e-06 | | -1.7054e+01 | | -2.3088e+01 | | -2.6665e+01 |
| 7.6250e-01 | | -5.0000e+07 | | 7.6250e-01 | | 7.0500e-01 | | 1.2532e-01 | | -1.2513e-01 | |
| | -1.6000e+01 | | -1.6000e+01 | | -1.7391e+01 | | -1.8779e+01 | | -2.7081e+01 | | |
| 7.0000e-01 | | 1.3750e-01 | | 4.3750e-02 | | -1.5441e-02 | | 4.8611e-03 | | | |
| | -1.7778e+01 | | -2.6667e+01 | | -3.4286e+01 | | 3.0476e+01 | | | | |
| 6.4375e-01 | | 2.5000e-02 | | -8.7500e-02 | | 1.6237e-15 | | | | | |
| | -1.9394e+01 | | -3.5556e+01 | | -2.2857e+01 | | | | | | |
| 5.9219e-01 | | -3.6875e-02 | | -8.7500e-03 | | | | | | | |
| | -2.0984e+01 | | 3.2000e-07 | | | | | | | | |
| 5.4453e-01 | | 1.0781e-02 | | | | | | | | | |
| | -2.2857e+01 | | | | | | | | | | |
| 5.0078e-01 | | | | | | | | | | | |

Table 2.3: Sequence (2.6): $\varepsilon$-array obtained by `EPSALGO` algorithm.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.6000e+00 | | | | | | | | | | | |
| | -2.5000e+00 | | | | | | | | | | |
| 1.2000e+00 | | 8.0000e-01 | | | | | | | | | |
| | -5.0000e+00 | | -2.1875e-07 | | | | | | | | |
| 1 .0000e+00 | | 1.0000e+00 | | 9.3333e-01 | | | | | | | |
| | -8.0000e+08 | | **-1.5000e+01** | | -1.2857e+01 | | | | | | |
| 1 .0000e+00 | | 1.0000e+00 | | 1.4000e+00 | | 6.6613e-16 | | | | | |
| | -1.0000e+01 | | -1.2500e+01 | | -1.3571e+01 | | -2.4344e+14 | | | | |
| 9.0000e-01 | | 6.0000e-01 | | 4.6667e-01 | | -3.4417e-15 | | -1.6024e-15 | | | |
| | -1.3333e+01 | | -2.0000e+01 | | -1.5714e+01 | | 3.0024e+14 | | 2.4851e+15 | | |
| 8.2500e-01 | | 4.5000e-01 | | 7.0000e-01 | | -1.1102e-16 | | -1.1447e-15 | | -1.3214e-15 | |
| | - 1.6000e+01 | | -1.6000e+01 | | -1.7143e+01 | | -6.6720e+14 | | -3.1746e+15 | | 4.6153e+15 |
| 7.6250e-01 | | -5.0000e+07 | | **-1.7500e-01** | | -1.6098e-15 | | -1.5435e-15 | | -1.1930e-15 | |
| | - 1.6000e+01 | | -1.6000e+01 | | -1.1429e+01 | | 1.4412e+16 | | -3.2157e+14 | | |
| 7.0000e-01 | | 1.3750e-01 | | 4.3750e-02 | | -1.5404e-15 | | -1.6114e-15 | | | |
| | -1.7778e+01 | | -2.6667e+01 | | -3.4286e+01 | | 3.1604e+14 | | | | |
| 6.4375e-01 | | 2.5000e-02 | | -8.7500e-02 | | 1.6237e-15 | | | | | |
| | -1.9394e+01 | | -3.5556e+01 | | -2.2857e+01 | | | | | | |
| 5.9219e-01 | | -3.6875e-02 | | -8.7500e-03 | | | | | | | |
| | -2.0984e+01 | | 3.2000e-07 | | | | | | | | |
| 5.4453e-01 | | 1.0781e-02 | | | | | | | | | |
| | -2.2857e+01 | | | | | | | | | | |
| 5.0078e-01 | | | | | | | | | | | |

Table 2.4: Sequence (2.6): $\varepsilon$-array obtained by `EPSALGOW` algorithm.

| $n$ | EPSALGO | EPSALGOW |
|---|---|---|
| 0 | 8.5200e-01 | 6.6613e-16 |
| 1 | -5.2500e-01 | -3.4417e-15 |
| 2 | 7.6364e-01 | -1.1102e-16 |
| 3 | 7.0500e-01 | -1.6098e-15 |
| 4 | -1.5441e-02 | -1.5404e-15 |
| 5 | 1.6237e-15 | 1.6237e-15 |

Table 2.5: Sequence (2.6). Values of $\varepsilon_6^{(n)}$, $n = 0, 1, \ldots 5$, as calculated by EPSALGO and EPSALGOW.

**Example 2.** [29, Annexe 4, p.43] We consider the sequence given by

$$\begin{aligned} &S_0 = S_1 = S_2 = 1,\ S_3 = 2, \\ &S_n = 3S_{n-4},\ \text{for n} \geq 4. \end{aligned} \tag{2.7}$$

Consulting Theorem 1.1, we expect that $\forall n$, $e_4(S_n) = 0$, thus $\varepsilon_8^{(n)} = 0$. The results obtained by EPSALGOC algorithm agree with this theory. In Table 2.6 apart from the values of $\varepsilon_8^{(n)}$ (coloured in red), we see in bold the values computed by the particular rules. In total eight singular cases are treated, namely six isolated singularities and two non-isolated singularities. Not all the singular square blocks are marked to avoid confusion, since some values are related to more than one blocks. For instance, the element $\varepsilon_5^{(2)} = 1.3333$ can be seen as the element $E_2$ related to the singularity caused by $\varepsilon_0^{(4)} = \varepsilon_0^{(5)} = \varepsilon_0^{(6)} = 3$ (non-isolated singularity), but at the same time together with the element $\varepsilon_5^{(3)}$ it causes an isolated singularity. We would like to stress that there are also two blocks that remain open, one caused by the elements $\varepsilon_0^{(0)} = \varepsilon_0^{(1)} = \varepsilon_0^{(2)} = 1$ (see Table 2.6) and another one due to the elements $\varepsilon_0^{(12)} = \varepsilon_0^{(13)} = \varepsilon_0^{(14)} = 27$. For these blocks there is no $E_i$ to be calculated, but they are treated as singular square blocks in the sense that we impose the value $\infty$ to the elements near the borders to avoid the propagation of the undefined value (NaN in Matlab notation) as explained in Remark 2.1.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | |
| | Inf | | | | | | | | | | | | | |
| 1 | | NaN | | | | | | | | | | | | |
| | Inf | | Inf | | | | | | | | | | | |
| 1 | | 1 | | 1 | | | | | | | | | | |
| | 1 | | 1 | | 2 | | | | | | | | | |
| 2 | | Inf | | **2** | | Inf | | | | | | | | |
| | 1 | | 1 | | 2 | | 2 | | | | | | | |
| 3 | | 3 | | 3 | | 1.5 | | 4.4409e-16 | | | | | | |
| | Inf | | Inf | | **1.3333e+00** | | 1.3333e+00 | | -2.2518e+15 | | | | | |
| 3 | | NaN | | 3 | | Inf | | **0** | | 0 | | | | |
| | Inf | | Inf | | **1.3333e+00** | | 1.3333e+00 | | Inf | | NaN | | | |
| 3 | | 3 | | 3 | | 1.5 | | 0 | | NaN | | NaN | | |
| | 3.3333e-01 | | 3.3333e-01 | | 6.6667e-01 | | 6.6667e-01 | | Inf | | NaN | | NaN | |
| 6 | | Inf | | **6** | | Inf | | **0** | | NaN | | NaN | | NaN |
| | 3.3333e-01 | | 3.3333e-01 | | 6.6667e-01 | | 6.6667e-01 | | Inf | | NaN | | NaN | |
| 9 | | 9 | | 9 | | 4.5 | | 0 | | NaN | | NaN | | |
| | Inf | | Inf | | **4.4444e-01** | | 4.4444e-01 | | Inf | | NaN | | | |
| 9 | | NaN | | 9 | | Inf | | **0** | | NaN | | | | |
| | Inf | | Inf | | **4.4444e-01** | | 4.4444e-01 | | Inf | | | | | |
| 9 | | 9 | | 9 | | 4.5 | | 0 | | | | | | |
| | 1.1111e-01 | | 1.1111e-01 | | 2.2222e-01 | | 2.2222e-01 | | | | | | | |
| 18 | | Inf | | **18** | | Inf | | | | | | | | |
| | 1.1111e-01 | | 1.1111e-01 | | 2.2222e-01 | | | | | | | | | |
| 27 | | 27 | | 27 | | | | | | | | | | |
| | Inf | | Inf | | | | | | | | | | | |
| 27 | | NaN | | | | | | | | | | | | |
| | Inf | | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | |

Table 2.6: Sequence (2.7): $\varepsilon$-array obtained by `EPSALGOC` algorithm.

**Example 3.** Now we will construct a sequence based on (2.6) and imposing the first three elements to be almost equal. In this way, using the $\varepsilon$-algorithm with $p = 8$ we meet several non-isolated singularities, which `EPSALGOW` cannot treat. In addition, the singularities are caused by almost equal values, which means that `EPSALGOC` algorithm fails.

So, let us consider the sequence given by

$$\begin{aligned} &S_0 = 1.599999999999,\ S_1 = 1.599999999995,\ S_2 = 1.599999999999, \\ &S_n = \frac{S_{n-1}}{2} + \frac{S_{n-2}}{4} + \frac{S_{n-3}}{8} \text{ for n} \geq 3. \end{aligned} \tag{2.8}$$

According to Theorem 1.1, $\forall n$, $\varepsilon_6^{(n)} = 0$, since $e_3(S_n) = \varepsilon_6^{(n)}$. Indeed, the corresponding column of the $\varepsilon$-array obtained by `EPSALGOG` algorithm gives values very close to zero (see the values in red in Table 2.9). In the contrary, if no particular rules are used, the results are not so good (see the values in red in Table 2.8). In Table 2.7 we compare the values of the elements $\varepsilon_6^{(n)}$, $n = 0, 1, \ldots 4$, as obtained by the two algorithms and we see that they only disagree in $\varepsilon_6^{(1)}$, $\varepsilon_6^{(2)}$. This comes as no surprise, since these two elements are those that are computed by the generalized particular rule when `EPSALGOG` algorithm is used. So before calculating $\varepsilon_6^{(1)}$, $\varepsilon_6^{(2)}$, all the values of the $\varepsilon$-array are the same whether we use `EPSALGO` algorithm or `EPSALGOG`. Of course, after the column where the particular rules are used for first time the results do not agree any more, although we use the normal rule to fill in the rest of the $\varepsilon$-array (provided that no other singularities are met).

| $n$ | `EPSALGO` | `EPSALGOG` |
|---|---:|---:|
| 0 | -5.5511e-16 | -5.5511e-16 |
| 1 | 2.8403e+06 | -8.0074e-12 |
| 2 | -2.8403e+06 | 8.0012e-12 |
| 3 | 6.6613e-16 | 6.6613e-16 |
| 4 | 3.2613e-16 | 3.2613e-16 |

Table 2.7: Sequence (2.8). Values of $\varepsilon_6^{(n)}$, $n = 0, 1, \ldots 4$, as calculated by `EPSALGO` and `EPSALGOG`.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1.6000e+00 | | | | | | | | | | |
| | -2.5001e+11 | | | | | | | | | |
| 1.6000e+00 | | 1.6000e+00 | | | | | | | | |
| | 2.5001e+11 | | -2.5001e+11 | | | | | | | |
| 1.6000e+00 | | 1.6000e+00 | | 1.6000e+00 | | | | | | |
| | -5.0000e+00 | | -7.5000e+00 | | -8.7500e+00 | | | | | |
| 1.4000e+00 | | 1.2000e+00 | | 8.0000e-01 | | -5.5511e-16 | | | | |
| | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | | |
| 1.3000e+00 | | -3.9996e+10 | | 3.9991e+10 | | 2.8403e+06 | | 9.4684e+05 | | |
| | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | |
| 1.2000e+00 | | 3.9996e+10 | | -3.9991e+10 | | -2.8403e+06 | | -9.4684e+05 | | -4.7344e+05 |
| | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | |
| 1.1000e+00 | | 4.0000e-01 | | 1.0000e-01 | | 6.6613e-16 | | 3.2613e-16 | | |
| | -1.1429e+01 | | -1.3333e+01 | | -2.0000e+01 | | -2.9411e+15 | | | |
| 1.0125e+00 | | -1.2500e-01 | | -5.0000e-02 | | 3.2613e-16 | | | | |
| | -1.2308e+01 | | 4.0367e-10 | | -4.0057e-10 | | | | | |
| 9.3125e-01 | | -4.3750e-02 | | -1.2434e+09 | | | | | | |
| | -1.3333e+01 | | -4.0057e-10 | | | | | | | |
| 8.5625e-01 | | 3.1250e-02 | | | | | | | | |
| | -1.4545e+01 | | | | | | | | | |
| 7.8750e-01 | | | | | | | | | | |

Table 2.8: Sequence (2.8): $\varepsilon$-array obtained by `EPSALGO` algorithm.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1.6000e+00 | | | | | | | | | | |
| | -2.5001e+11 | | | | | | | | | |
| 1.6000e+00 | | 1.6000e+00 | | | | | | | | |
| | 2.5001e+11 | | -2.5001e+11 | | | | | | | |
| 1.6000e+00 | | 1.6000e+00 | | 1.6000e+00 | | | | | | |
| | -5.0000e+00 | | -7.5000e+00 | | -8.7500e+00 | | | | | |
| 1.4000e+00 | | 1.2000e+00 | | 8.0000e-01 | | -5.5511e-16 | | | | |
| | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | -1.2489e+11 | | | |
| 1.3000e+00 | | -3.9996e+10 | | 3.9991e+10 | | **-8.0074e-12** | | -2.6701e-12 | | |
| | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | 6.2467e+10 | | 2.4985e+11 | |
| 1.2000e+00 | | 3.9996e+10 | | -3.9991e+10 | | **8.0012e-12** | | 2.6667e-12 | | 1.3331e-12 |
| | -1.0000e+01 | | -1.0000e+01 | | -1.0000e+01 | | -1.2499e+11 | | -5.0004e+11 | |
| 1.1000e+00 | | 4.0000e-01 | | 1.0000e-01 | | 6.6613e-16 | | 3.2611e-16 | | |
| | -1.1429e+01 | | -1.3333e+01 | | -2.0000e+01 | | -2.9411e+15 | | | |
| 1.0125e+00 | | -1.2500e-01 | | -5.0000e-02 | | 3.2613e-16 | | | | |
| | -1.2308e+01 | | 4.0367e-10 | | -4.0057e-10 | | | | | |
| 9.3125e-01 | | -4.3750e-02 | | -1.2434e+09 | | | | | | |
| | -1.3333e+01 | | -4.0057e-10 | | | | | | | |
| 8.5625e-01 | | 3.1250e-02 | | | | | | | | |
| | -1.4545e+01 | | | | | | | | | |
| 7.8750e-01 | | | | | | | | | | |

Table 2.9: Sequence (2.8): $\varepsilon$-array obtained by EPSALGOG algorithm.

**Example 4.** The last test sequence is the next one

$$\begin{aligned} &S_0 = 0.999999999999,\ S_1 = 1,\ S_2 = 1.000000000001,\ S_3 = 1.5 \\ &S_n = 3S_{n-4} \text{ for n} \geq 4. \end{aligned} \tag{2.9}$$

According to Theorem 1.1, $\forall n$, $e_4(S_n) = 0$ , thus $\varepsilon_8^{(n)} = 0$. As we can see in Table 2.10, the results of `EPSALGO` algorithm are far from the theoretical results, whilst `EPSALGOG` algorithm gives values very close to zero. In Tables 2.11 and 2.12 one can see the whole $\varepsilon$-array resulting from the two algorithms. In Table 2.12 the two singular square blocks (both of size 3) are drawn and the values computed by the particular rules are written in bold. We stress that these values correspond to an odd column, thus they are intermediate values. However, they contribute via the normal rule to the better results of column $\varepsilon_8$ in which we are interested.

| $n$ | `EPSALGO` | `EPSALGOG` |
|---|---:|---:|
| 0 | 1.9587e+00 | 1.0658e-14 |
| 1 | Inf | -8.8818e-16 |
| 2 | 4.3333e+00 | -1.1102e-14 |
| 3 | -4.0343e+00 | 1.0658e-14 |
| 4 | 5.8761e+00 | -3.1974e-14 |
| 5 | 7.9092e+00 | 1.7764e-15 |

Table 2.10: Sequence (2.9). Values of $\varepsilon_8^{(n)}$, $n = 0, 1, \ldots 5$, as calculated by `EPSALGO` and `EPSALGOG`.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.0000e+00 | | | | | | | | | | | | | |
| | 1.0000e+12 | | | | | | | | | | | | |
| 1.0000e+00 | | 1.0000e+00 | | | | | | | | | | | |
| | 9.9991e+11 | | 1.0000e+12 | | | | | | | | | | |
| 1.0000e+00 | | 1.0000e+00 | | 1.0000e+00 | | | | | | | | | |
| | 2.0000e+00 | | -2.0000e+00 | | 1.2000e+01 | | | | | | | | |
| 1.5000e+00 | | 7.5000e-01 | | 1.0714e+00 | | 9.7500e-01 | | | | | | | |
| | 6.6667e-01 | | 1.1111e+00 | | 1.6296e+00 | | 2.3382e+00 | | | | | | |
| 3.0000e+00 | | 3.0000e+00 | | 3.0000e+00 | | 2.3864e+00 | | 1.9587e+00 | | | | | |
| | 3.3335e+11 | | 3.3330e+11 | | 0 | | 0 | | 0 | | | | |
| 3.0000e+00 | | 3.0000e+00 | | 3.0000e+00 | | Inf | | Inf | | Inf | | | |
| | 3.3330e+11 | | 3.3335e+11 | | 0 | | 0 | | 0 | | 0 | | |
| 3.0000e+00 | | 3.0000e+00 | | 3.0000e+00 | | 3.2500e+00 | | 4.3333e+00 | | 5.5778e+00 | | 6.4904e+00 | |
| | 6.6667e-01 | | -6.6667e-01 | | 4.0000e+00 | | 9.2308e-01 | | 8.0357e-01 | | 1.0958e+00 | | -1.9712e+01 |
| 4.5000e+00 | | 2.2500e+00 | | 3.2143e+00 | | 2.9250e+00 | | -4.0343e+00 | | 9.0001e+00 | | 6.4423e+00 | |
| | 2.2222e-01 | | 3.7037e-01 | | 5.4321e-01 | | 7.7939e-01 | | 8.8029e-01 | | 7.0480e-01 | | |
| 9.0000e+00 | | 9.0000e+00 | | 9.0000e+00 | | 7.1591e+00 | | 5.8761e+00 | | 3.3016e+00 | | | |
| | 1.1112e+11 | | 1.1110e+11 | | -1.5259e-05 | | 1.5257e-05 | | 4.9186e-01 | | | | |
| 9.0000e+00 | | 9.0000e+00 | | 9.0000e+00 | | 3.2777e+04 | | 7.9092e+00 | | | | | |
| | 1.1110e+11 | | 1.1112e+11 | | 1.5259e-05 | | -1.5259e-05 | | | | | | |
| 9.0000e+00 | | 9.0000e+00 | | 9.0000e+00 | | 9.7500e+00 | | | | | | | |
| | 2.2222e-01 | | -2.2222e-01 | | 1.3333e+00 | | | | | | | | |
| 1.3500e+01 | | 6.7500e+00 | | 9.6429e+00 | | | | | | | | | |
| | 7.4074e-02 | | 1.2346e-01 | | | | | | | | | | |
| 2.7000e+01 | | 2.7000e+01 | | | | | | | | | | | |
| | 3.7041e+10 | | | | | | | | | | | | |
| 2.7000e+01 | | | | | | | | | | | | | |

Table 2.11: Sequence (2.9): $\varepsilon$-array obtained by `EPSALGO` algorithm.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.0000e+00 | | | | | | | | | | | | | |
| | 1.0000e+12 | | | | | | | | | | | | |
| 1.0000e+00 | | 1.0000e+00 | | | | | | | | | | | |
| | 9.9991e+11 | | 1.0000e+12 | | | | | | | | | | |
| 1.0000e+00 | | 1.0000e+00 | | 1.0000e+00 | | | | | | | | | |
| | 2.0000e+00 | | -2.0000e+00 | | 1.2000e+01 | | | | | | | | |
| 1.5000e+00 | | 7.5000e-01 | | 1.0714e+00 | | 9.7500e-01 | | | | | | | |
| | 6.6667e-01 | | 1.1111e+00 | | 1.6296e+00 | | 1.7436e+00 | | | | | | |
| 3.0000e+00 | | 3.0000e+00 | | 3.0000e+00 | | 9.7500e+00 | | 1.0658e-14 | | | | | |
| | 3.3335e+11 | | 3.3330e+11 | | **1.7778e+00** | | 1.6410e+00 | | -8.6608e+13 | | | | |
| 3.0000e+00 | | 3.0000e+00 | | 3.0000e+00 | | 2.4375e+00 | | -8.8818e-16 | | -8.9410e-14 | | | |
| | 3.3330e+11 | | 3.3335e+11 | | **-2.6683e-12** | | 1.2308e+00 | | -9.7904e+13 | | -8.6175e+13 | | |
| 3.0000e+00 | | 3.0000e+00 | | 3.0000e+00 | | 3.2500e+00 | | -1.1102e-14 | | -4.1510e-15 | | -3.7690e-15 | |
| | 6.6667e-01 | | -6.6667e-01 | | 4.0000e+00 | | 9.2308e-01 | | 4.5955e+13 | | 2.5318e+15 | | -2.8204e+14 |
| 4.5000e+00 | | 2.2500e+00 | | 3.2143e+00 | | 2.9250e+00 | | 1.0658e-14 | | -3.7487e-15 | | -4.1244e-15 | |
| | 2.2222e-01 | | 3.7037e-01 | | 5.4321e-01 | | 5.8120e-01 | | -2.3456e+13 | | -1.2997e+14 | | |
| 9.0000e+00 | | 9.0000e+00 | | 9.0000e+00 | | 2.9250e+01 | | -3.1974e-14 | | -1.3137e-14 | | | |
| | 1.1112e+11 | | 1.1110e+11 | | **5.9259e-01** | | 5.4701e-01 | | 2.9629e+13 | | | | |
| 9.0000e+00 | | 9.0000e+00 | | 9.0000e+00 | | 7.3125e+00 | | 1.7764e-15 | | | | | |
| | 1.1110e+11 | | 1.1112e+11 | | **-8.8954e-13** | | 4.1026e-01 | | | | | | |
| 9.0000e+00 | | 9.0000e+00 | | 9.0000e+00 | | 9.7500e+00 | | | | | | | |
| | 2.2222e-01 | | -2.2222e-01 | | 1.3333e+00 | | | | | | | | |
| 1.3500e+01 | | 6.7500e+00 | | 9.6429e+00 | | | | | | | | | |
| | 7.4074e-02 | | 1.2346e-01 | | | | | | | | | | |
| 2.7000e+01 | | 2.7000e+01 | | | | | | | | | | | |
| | 3.7041e+10 | | | | | | | | | | | | |
| 2.7000e+01 | | | | | | | | | | | | | |

Table 2.12: Sequence (2.9): $\varepsilon$-array obtained by `EPSALGOG` algorithm.

# Chapter 3

# Generalizations of Aitken's process

This chapter presents the work originally developed in the published paper [23]. We study the class of sequences

$$S_n = S + a_n \lambda^n, \quad n = 0, 1, \ldots, \tag{3.1}$$

where $S$ and $\lambda$ are unknown parameters, and $(a_n)$ is a known sequence, which generalizes the kernel of Aitken's process (1.2). We propose three transformations whose kernel contains sequences of the form (3.1), thus can be considered generalizations of Aitken's $\Delta^2$ process which we studied in Section 1.2. For one of the proposed transformations we prove the convergence acceleration, under certain sufficient conditions, of certain types of sequences. Numerical results including comparisons with other transformations, and experiments using both divergent and convergent sequences conclude the chapter.

We recall that Aitken's process (1.2) converges if and only if $|\lambda| \leq 1$. However, the convergence of sequences of the form (3.1) depends on the convergence of the term $a_n \lambda^n$, thus both on the behavior of $(a_n)$ and on $\lambda$. For instance, if $(a_n)$ is mildly increasing, (3.1) converges only for values $|\lambda| < 1$.

## 3.1 New generalizations of Aitken's $\Delta^2$ process

The technique we follow in order to derive the generalizations of Aitken's $\Delta^2$ process is similar to the one presented in Section 1.2.Starting from the kernel (3.1), we obtain $S$ as a function, namely $S = f(S_n, \ldots, S_{n+k}; a_n, \ldots, a_{n+l})$ for some $k, l \in \mathbb{N}$ and for all $n$, we compute $a_n, \ldots, a_{n+l}$ by an interpolation process depending on $n$, and we define the transformation as $T_n := f(S_n, \ldots, S_{n+k}; a_n, \ldots, a_{n+l})$ for $n = 0, 1, \ldots$ .

**Transformation ${}_1T_n$.** As we already mentioned the starting point is the kernel. Writing (3.1) for the indices $n$ and $n+1$ , we have

$$\left\{ \begin{array}{rcl} S_n & = & S + a_n\lambda^n \\ S_{n+1} & = & S + a_{n+1}\lambda^{n+1}. \end{array} \right. \tag{3.2}$$

The first equation gives $\lambda^n = (S_n - S)/a_n$, thus the second equation becomes

$$a_nS - a_{n+1}\lambda S + a_{n+1}S_n\lambda = a_nS_{n+1}. \tag{3.3}$$

Instead of solving the nonlinear equation (3.3) with unknowns $S$ and $\lambda$, we rewrite it for the indices $n, n+1, n+2$. So we end up with a system of three linear equations in the three unknowns $S, \lambda S, \lambda$

$$\left\{ \begin{array}{rcl} a_nS - a_{n+1}\lambda S + a_{n+1}S_n\lambda & = & a_nS_{n+1} \, , \\ a_{n+1}S - a_{n+2}\lambda S + a_{n+2}S_{n+1}\lambda & = & a_{n+1}S_{n+2} \, , \\ a_{n+2}S - a_{n+3}\lambda S + a_{n+3}S_{n+2}\lambda & = & a_{n+2}S_{n+3} \, . \end{array} \right. \tag{3.4}$$

Computing $S$ directly from (3.4) as a ratio of determinants we obtain our first transformation

$${}_1T_n = \frac{\begin{vmatrix} a_nS_{n+1} & -a_{n+1} & a_{n+1}S_n \\ a_{n+1}S_{n+2} & -a_{n+2} & a_{n+2}S_{n+1} \\ a_{n+2}S_{n+3} & -a_{n+3} & a_{n+3}S_{n+2} \end{vmatrix}}{\begin{vmatrix} a_n & -a_{n+1} & a_{n+1}S_n \\ a_{n+1} & -a_{n+2} & a_{n+2}S_{n+1} \\ a_{n+2} & -a_{n+3} & a_{n+3}S_{n+2} \end{vmatrix}} = \frac{\begin{vmatrix} a_{n+1} & a_{n+2} & a_{n+3} \\ a_nS_{n+1} & a_{n+1}S_{n+2} & a_{n+2}S_{n+3} \\ a_{n+1}S_n & a_{n+2}S_{n+1} & a_{n+3}S_{n+2} \end{vmatrix}}{\begin{vmatrix} a_{n+1} & a_{n+2} & a_{n+3} \\ a_n & a_{n+1} & a_{n+2} \\ a_{n+1}S_n & a_{n+2}S_{n+1} & a_{n+3}S_{n+2} \end{vmatrix}} = \frac{N_n}{D_n} \tag{3.5}$$

The numerator $N_n$ and the denominator $D_n$ can be expressed in a compact form as follows

$$\begin{array}{rcl} N_n & = & a_{n+3}\Delta S_{n+1}(a_{n+1}^2S_{n+2} - a_na_{n+2}S_{n+1}) - a_{n+1}\Delta S_n(a_{n+2}^2S_{n+3} - a_{n+1}a_{n+3}S_{n+2}) \, , \\ D_n & = & a_{n+3}\Delta S_{n+1}(a_{n+1}^2 - a_na_{n+2}) - a_{n+1}\Delta S_n(a_{n+2}^2 - a_{n+1}a_{n+3}) \end{array}$$

As we explained in Section 1.2, a transformation written in the preceding form is unstable. Nevertheless, since the computation of ${}_1T_n$ uses the terms $S_n, S_{n+1}, S_{n+2}$ and

$S_{n+3}$, we may write it as ${}_1T_n = S_{n+i} - (S_{n+i}D_n - N_n)/D_n$, $i = 0, 1, 2, 3$, thus obtain the following more stable formulas

$$S_n - \frac{\big(a_n\, a_{n+2}\, a_{n+3}\, \Delta S_{n+1} + a_{n+1}\, a_{n+2}^2\, (S_{n+3} - S_n)\big)\, \Delta S_n - a_{n+1}^2\, a_{n+3}\, (\Delta S_{n+1} + \Delta S_n)^2}{D_n}, \tag{3.6}$$

$$S_{n+1} - \frac{a_{n+1}\, a_{n+2}^2\, \Delta S_n\, (\Delta S_{n+2} + \Delta S_{n+1}) - a_{n+1}^2\, a_{n+3}\, \Delta S_{n+1}\, (\Delta S_{n+1} + \Delta S_n)}{D_n}, \tag{3.7}$$

$$S_{n+2} - \frac{a_{n+1}\, a_{n+2}^2\, \Delta S_n \Delta S_{n+2} - a_n\, a_{n+2}\, a_{n+3}\, (\Delta S_{n+1})^2}{D_n}, \tag{3.8}$$

$$S_{n+3} - \frac{a_{n+1}^2\, a_{n+3}\, \Delta S_{n+2}\, (\Delta S_{n+1} + \Delta S_n) - a_n\, a_{n+2}\, a_{n+3}\, \Delta S_{n+1}\, (\Delta S_{n+1} + \Delta S_{n+2})}{D_n}. \tag{3.9}$$

Assuming $a_n \neq 0$ for all $n \in \mathbb{N}_0$, we can derive another equivalent expression for the transformation ${}_1T_n$. First we divide the $i$th column in both the numerator and the denominator of (3.5) by $a_{n+i}$. Next we replace the second and third column by their difference with the preceding ones, reducing $N_n$ and $D_n$ into $2 \times 2$ determinants. By setting $\beta_n = a_n/a_{n+1}$ we obtain

$$ {}_1T_n \quad = \quad \frac{\begin{vmatrix} \Delta\,(\beta_n\, S_{n+1}) & \Delta\,(\beta_{n+1}\, S_{n+2}) \\ \Delta S_n & \Delta S_{n+1} \end{vmatrix}}{\begin{vmatrix} \Delta\beta_n & \Delta\beta_{n+1} \\ \Delta S_n & \Delta S_{n+1} \end{vmatrix}}\,. $$

Then, assuming that $\Delta S_n \neq 0$ for all $n \in \mathbb{N}_0$, we divide the $i$th column by $\Delta S_{n+i-1}$, $i = 1, 2$, thus we have a compact form of our first transformation

$$ {}_1T_n \quad = \quad \frac{\Delta\left(\dfrac{\Delta\,(\beta_n\, S_{n+1})}{\Delta S_n}\right)}{\Delta\left(\dfrac{\Delta\beta_n}{\Delta S_n}\right)}\,. $$

Now, we can derive more equivalent formulas for ${}_1T_n$, analogous to (3.6-3.9), i.e.

$$
\begin{aligned}
{}_1T_n &= S_n + \frac{\Delta^2\beta_n + \Delta\beta_{n+1}\dfrac{\Delta S_n}{\Delta S_{n+1}} + \Delta\left(\beta_{n+1}\dfrac{\Delta S_{n+1}}{\Delta S_n}\right)}{\Delta\left(\dfrac{\Delta\beta_n}{\Delta S_n}\right)} \\
&= S_{n+1} + \frac{\Delta\beta_{n+1} + \Delta\left(\beta_{n+1}\dfrac{\Delta S_{n+1}}{\Delta S_n}\right)}{\Delta\left(\dfrac{\Delta\beta_n}{\Delta S_n}\right)} \\
&= S_{n+2} + \frac{\Delta\beta_{n+1}\dfrac{\Delta S_{n+2}}{\Delta S_{n+1}} + \Delta\left(\beta_n\dfrac{\Delta S_{n+1}}{\Delta S_n}\right)}{\Delta\left(\dfrac{\Delta\beta_n}{\Delta S_n}\right)} \\
&= S_{n+3} + \frac{\Delta\beta_n\dfrac{\Delta S_{n+2}}{\Delta S_n} + \Delta\left(\beta_n\dfrac{\Delta S_{n+1}}{\Delta S_n}\right)}{\Delta\left(\dfrac{\Delta\beta_n}{\Delta S_n}\right)}\,.
\end{aligned}
$$

**Remark 3.1.** *If $a_n = a$ for all $n \in \mathbb{N}_0$, then* (3.1) *reduces to* (1.2). *In this case the system* (3.4) *has more than one solution, that we can express as follows*

$$
\begin{pmatrix} S \\ \lambda S \\ \lambda \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} S_{n+1} \\ S_n\Delta S_{n+1}/\Delta S_n \\ \Delta S_{n+1}/\Delta S_n \end{pmatrix}, \ \textit{for } \alpha \in \mathbb{R}. \tag{3.10}
$$

*The first element of this vector is $S \equiv {}_1T_n$ and if we set $\alpha = -\dfrac{\Delta S_n \Delta S_{n+1}}{\Delta^2 S_n}$, we obtain the second expression of* (1.11).

**Transformation ${}_2T_n$.** Our second sequence transformation results directly from the system (3.2), which under the hypothesis that $\beta_n \neq \lambda$, gives $S$. Taking $T_n = S$ we have the transformation

$$
T_n = \frac{a_{n+1}S_n\lambda - a_nS_{n+1}}{a_{n+1}\lambda - a_n}\,, \tag{3.11}
$$

that can be also written in a more stable form as follows

$$\begin{aligned} T_n &= S_n - \frac{a_n\,\Delta S_n}{a_{n+1}\,\lambda - a_n} = S_n - \frac{\beta_n\,\Delta S_n}{\lambda - \beta_n} \\ &= S_{n+1} - \frac{a_{n+1}\,\Delta S_n \lambda}{a_{n+1}\,\lambda - a_n} = S_{n+1} - \frac{\Delta S_n\,\lambda}{\lambda - \beta_n}\,. \end{aligned} \tag{3.12}$$

In order to define our second transformation we need to compute the unknown $\lambda$. One way is to solve the system (3.4). Using this $\lambda$ in (3.12), gives a transformation denoted in the sequel as ${}_2T_n$. Solving system (3.4), we obtain

$$\lambda = \frac{\begin{vmatrix} a_n & -a_{n+1} & a_n\,S_{n+1} \\ a_{n+1} & -a_{n+2} & a_{n+1}\,S_{n+2} \\ a_{n+2} & -a_{n+3} & a_{n+2}\,S_{n+3} \end{vmatrix}}{\begin{vmatrix} a_n & -a_{n+1} & a_{n+1}\,S_n \\ a_{n+1} & -a_{n+2} & a_{n+2}\,S_{n+1} \\ a_{n+2} & -a_{n+3} & a_{n+3}\,S_{n+2} \end{vmatrix}} = \frac{\begin{vmatrix} a_{n+1} & a_{n+2} & a_{n+3} \\ a_n & a_{n+1} & a_{n+2} \\ a_nS_{n+1} & a_{n+1}S_{n+2} & a_{n+2}S_{n+3} \end{vmatrix}}{\begin{vmatrix} a_{n+1} & a_{n+2} & a_{n+3} \\ a_n & a_{n+1} & a_{n+2} \\ a_{n+1}S_n & a_{n+2}S_{n+1} & a_{n+3}S_{n+2} \end{vmatrix}}\,. \tag{3.13}$$

Assuming that $a_n \neq 0$ and $\Delta\beta_n \neq 0$ for all $n \in \mathbb{N}_0$, we perform analogous algebraic manipulations as done in (3.5) and we arrive at

$$\lambda = \frac{\Delta\left(\dfrac{\Delta\left(\beta_n\,S_{n+1}\right)}{\Delta\beta_n}\right)}{\Delta\left(\dfrac{\Delta S_n}{\Delta\beta_n}\right)}\,. \tag{3.14}$$

One can easily confirm that, if $\beta_n$ converges to $\beta \in \mathbb{R}$, with $\beta \neq \lambda$, then $\beta_{n+1}\dfrac{\Delta S_{n+1}}{\Delta S_n}$ converges to $\lambda$. Hence, we can rewrite (3.14) in the following way

$$\lambda = \beta_{n+2}\frac{\Delta S_{n+2}}{\Delta S_{n+1}} + \frac{\Delta S_n\,\Delta\left(\beta_n\beta_{n+1}\dfrac{\Delta S_{n+1}}{\Delta S_n}\right)}{\beta_{n+1}\,\Delta\beta_n\,\Delta\left(\dfrac{\Delta S_n}{\Delta\beta_n}\right)}\,.$$

thus

$$\begin{aligned} {}_2T_n &= S_n - \frac{\beta_n\, \Delta S_n \Delta\left(\dfrac{\Delta S_n}{\Delta\beta_n}\right)}{\Delta\left(\dfrac{\Delta\left(\beta_n\, S_{n+1}\right)}{\Delta\beta_n}\right) - \beta_n\,\Delta\left(\dfrac{\Delta S_n}{\Delta\beta_n}\right)} \\ &= S_{n+1} - \frac{\Delta S_n\, \Delta\left(\dfrac{\Delta\left(\beta_n\, S_{n+1}\right)}{\Delta\beta_n}\right)}{\Delta\left(\dfrac{\Delta\left(\beta_n\, S_{n+1}\right)}{\Delta\beta_n}\right) - \beta_n\Delta\left(\dfrac{\Delta S_n}{\Delta\beta_n}\right)}\,. \end{aligned}$$

**Remark 3.2.** *We already saw in Remark 3.1 that, if* $a_n = a$ *for all* $n \in \mathbb{N}_0$*, the solutions of the system* (3.4) *are expressed in the equation* (3.10)*, which gives for* $\lambda$ *the value* $\dfrac{\Delta S_{n+1}}{\Delta S_n}, \forall\alpha \in \mathbb{R}$*. Then,* (3.12) *becomes equal to* (1.11)*, that is from* ${}_2T_n$ *we recover Aitken's* $\Delta^2$ *process.*

**Transformation** ${}_3T_n$**.** Another way to compute $\lambda$ in (3.12) is to apply the forward difference operator $\Delta$ to the system of equations (3.2), that is

$$\left\{ \begin{array}{rcl} \Delta S_n &=& \lambda^n(a_{n+1}\lambda - a_n)\,, \\ \Delta S_{n+1} &=& \lambda^{n+1}(a_{n+2}\lambda - a_{n+1})\,. \end{array} \right.$$

We eliminate the unknowns $\lambda^n, \lambda^{n+1}$ by division, thus we end up with a quadratic equation for the unknown $\lambda$

$$a_{n+2}\Delta S_n\lambda^2 - a_{n+1}(\Delta S_n + \Delta S_{n+1})\lambda + a_n\Delta S_{n+1} = 0\,. \tag{3.15}$$

This equation provides two solutions for $\lambda$ and the criterion according to which we accept one solution and reject the other one, is based on $\lambda$. For this reason, we write (3.15) for the indices $n$ and $n+1$ and we consider the following system in the two unknowns $\lambda$ and $\lambda^2$

$$\left\{ \begin{array}{rcl} a_{n+1}(\Delta S_n + \Delta S_{n+1})\lambda - a_{n+2}\Delta S_n\lambda^2 &=& a_n\Delta S_{n+1}\,, \\ a_{n+2}(\Delta S_{n+1} + \Delta S_{n+2})\lambda - a_{n+3}\Delta S_{n+1}\lambda^2 &=& a_{n+1}\Delta S_{n+2}\,. \end{array} \right. \tag{3.16}$$

We solve the system for $\lambda$ and by using this value of $\lambda$ in (3.12) we obtain transformation ${}_3T_n$.

An explicit expression for $\lambda$ is

$$\lambda \;=\; \frac{\Delta\left(\beta_n\beta_{n+1}\dfrac{\Delta S_{n+1}}{\Delta S_n}\right)}{\Delta\left(\beta_{n+1}\dfrac{\Delta S_n+\Delta S_{n+1}}{\Delta S_n}\right)}$$

and transformation ${}_3T_n$ can be expressed in one of the following equivalent formulas

$$\begin{aligned} {}_3T_n \;&=\; S_n+\beta_n\frac{\Delta S_n}{\Delta S_{n+2}}\frac{\Delta\left(\beta_{n+1}\dfrac{\Delta S_n+\Delta S_{n+1}}{\Delta S_n}\right)}{\beta_n\dfrac{\Delta\beta_{n+1}}{\Delta S_{n+2}}-\beta_{n+2}\dfrac{\Delta\beta_n}{\Delta S_{n+1}}} \\ &=\; S_{n+1}+\frac{\Delta S_n}{\Delta S_{n+2}}\frac{\Delta\left(\beta_n\beta_{n+1}\dfrac{\Delta S_{n+1}}{\Delta S_n}\right)}{\beta_n\dfrac{\Delta\beta_{n+1}}{\Delta S_{n+2}}-\beta_{n+2}\dfrac{\Delta\beta_n}{\Delta S_{n+1}}}\,. \end{aligned}$$

**Remark 3.3.** *A result analogous to that stated for transformations ${}_1T_n$ and ${}_2T_n$ in Remarks 3.1 and 3.2, holds for transformation ${}_3T_n$. It suffices to prove that for $a_n$ constant, $\lambda = \dfrac{\Delta S_{n+1}}{\Delta S_n}$ is a solution of the system* (3.16). *In fact, we can write the solutions of this system as*

$$\begin{pmatrix}\lambda\\ \lambda^2\end{pmatrix} = \alpha\begin{pmatrix}\Delta S_n\\ \Delta S_n+\Delta S_{n+1}\end{pmatrix}+\begin{pmatrix}1\\ 1\end{pmatrix},\ \textit{for } \alpha\in\mathbb{R}.$$

*Aitken's $\Delta^2$ process is recovered for $\alpha = \dfrac{\Delta^2 S_n}{(\Delta S_n)^2}$.*

## 3.2 Convergence and acceleration properties

Our numerical test (cf. Section 3.3.1) show that, among the three transformations presented in the previous section, the one that performs best is ${}_2T_n$. For this reason, we will focus on this one and study its convergence and acceleration properties.
Let us consider the sequence

$$\tilde{S}_n = S + a_n\lambda^n + g_n, \quad n = 0,1,\ldots, \tag{3.17}$$

where $g_n$ is a "noise term", in the sense that $\tilde{S}_n$ is not "too far" from the kernel (3.1). In order to prove some results on the convergence of transformation ${}_2T_n$, we will make several assumptions on the convergence behavior of $g_n$ and $a_n$ as $n \to \infty$. We first recall a well-known criterion for the characterization of the convergence. We assume that the elements of the sequence (3.17) satisfy

$$\lim_{n\to\infty} \frac{\tilde{S}_{n+1} - S}{\tilde{S}_n - S} = \rho.$$

We define the convergence rate of the sequence as the quantity $|\rho|$. If $0 < |\rho| < 1$ then $\tilde{S}_n$ converges linearly, if $\rho = 1$ then the convergence is logarithmic, whilst $\rho = 0$ means superlinear convergence. Finally, $|\rho| > 1$ implies divergence.

From now on we will assume that $g_n$ is subdominant to $a_n\lambda^n$ as $n \to \infty$, i.e. $\lim\limits_{n\to\infty} g_n/(a_n\lambda^n) = 0$. In particular, this means that the convergence rate of $\tilde{S}_n$ depends only on the term $a_n\lambda^n$.

Let $\beta = \lim\limits_{n\to\infty} \beta_n = \lim\limits_{n\to\infty} a_n/a_{n+1}$. If $|\beta|$ exists and is finite then

$$\lim_{n\to\infty} \frac{\tilde{S}_{n+1} - S}{\tilde{S}_n - S} = \frac{\lambda}{\beta}.$$

If $|\lambda| < |\beta|$ then $\tilde{S}_n$ converges linearly; for $\lambda = \beta$ we have logarithmic convergence; if $|\lambda| > |\beta|$ the sequence diverges. Finally, if $|\beta| = \infty$ then $\tilde{S}_n$ is superlinearly convergent for every value of $\lambda$. In this case convergence acceleration techniques are not really useful unless $|\lambda|$ is sufficiently large. For that reason we exclude the case in which $|\beta| = \infty$, in order to achieve some theoretical results on the acceleration of ${}_2T_n$.

Note that for the sequence (3.17) the value of $\lambda$ given from the system (3.4) is an approximation depending on $n$, let denote it as $\lambda_n$. Analogously to (3.14), for the approximation $\lambda_n$ we have the following formula

$$\lambda_n = \frac{\Delta\left(\dfrac{\Delta\left(\beta_n \tilde{S}_{n+1}\right)}{\Delta\beta_n}\right)}{\Delta\left(\dfrac{\Delta\tilde{S}_n}{\Delta\beta_n}\right)}. \tag{3.18}$$

Let us study the convergence of $\lambda_n$ as $n \to \infty$. First, we give the following technical lemmas.

**Lemma 3.1.** *If $g_n/(a_n\lambda^n) \to 0$ and $\beta_n$ is bounded, then $\Delta g_n/(a_{n+1}\lambda^n) \to 0$.*

**Proof.**

$$\lim_{n\to\infty} \frac{\Delta g_n}{a_{n+1}\lambda^n} = \lim_{n\to\infty}\left(\frac{g_{n+1}}{a_{n+1}\lambda^n} - \frac{g_n}{a_{n+1}\lambda^n}\right) = \lim_{n\to\infty}\left(\lambda\frac{g_{n+1}}{a_{n+1}\lambda^{n+1}} - \frac{g_n}{a_n\lambda^n}\beta_n\right) = \lambda 0 - 0 = 0.$$

□

**Lemma 3.2.** *Consider the sequence* (3.17) *and assume that*

1. $\displaystyle\lim_{n\to\infty}\frac{g_n}{a_n\lambda^n} = 0\,,$

2. *there exists a finite number $\beta$ such that $\beta \neq \lambda$ for which* $\displaystyle\lim_{n\to\infty}\beta_n = \lim_{n\to\infty}\frac{a_n}{a_{n+1}} = \beta\,.$

*Then* $\displaystyle\lim_{n\to\infty}\beta_{n+1}\frac{\Delta\tilde{S}_{n+1}}{\Delta\tilde{S}_n} = \lambda\,.$

**Proof.** Making use of the hypotheses 1 and 2 we have

$$\begin{aligned}\lim_{n\to\infty}\beta_{n+1}\frac{\Delta\tilde{S}_{n+1}}{\Delta\tilde{S}_n} &= \lim_{n\to\infty}\frac{a_{n+1}}{a_{n+2}}\frac{a_{n+2}\lambda^{n+1}\left(\lambda - \dfrac{a_{n+1}}{a_{n+2}} + \dfrac{\Delta g_{n+1}}{a_{n+2}\lambda^{n+1}}\right)}{a_{n+1}\lambda^n\left(\lambda - \dfrac{a_n}{a_{n+1}} + \dfrac{\Delta g_n}{a_{n+1}\lambda^n}\right)} \\ &= \lim_{n\to\infty}\lambda\frac{\lambda - \beta_{n+1} + \dfrac{\Delta g_{n+1}}{a_{n+2}\lambda^{n+1}}}{\lambda - \beta_n + \dfrac{\Delta g_n}{a_{n+1}\lambda^n}} = \lambda\frac{\lambda-\beta+0}{\lambda-\beta+0} = \lambda\,,\end{aligned}$$

since $\dfrac{\Delta g_n}{a_{n+1}\lambda^n}$ and $\dfrac{\Delta g_{n+1}}{a_{n+2}\lambda^{n+1}}$ converge to zero by Lemma 3.1.

□

We underline the meaning of hypothesis 1; it implies that the sequence is "not too far" from the kernel of the transformation. If $\beta = \lambda$ then $\tilde{S}_n$ converges logarithmically. The case in which $\beta$ does not exist is not easily interpreted. In Section 3.3.4 we discuss several numerical examples related to these particular situations.

We introduce the sequence $\gamma_n = \dfrac{a_{n+2}^2 - a_{n+1}\,a_{n+3}}{a_{n+1}^2 - a_n\,a_{n+2}}$, that we will use in the theorem on the convergence of $(\lambda_n)$.

**Theorem 3.1.** *The sequence $(\lambda_n)$ converges to $\lambda$ if the following conditions are satisfied:*

1. $\lim_{n\to\infty} \dfrac{g_n}{a_n\,\lambda^n} = 0\,,$
2. *there exists* $\beta \in \mathbb{R}$ *such that* $\lim_{n\to\infty} \beta_n = \beta\,,$
3. *there exists* $\gamma \in \mathbb{R}$ *such that* $\lim_{n\to\infty} \gamma_n = \gamma\,,$
4. $\lambda$, $\beta$ *and* $\gamma$ *are such that* $\beta \neq \lambda$ *and* $\lambda - \beta^3\gamma \neq 0$ .

**Proof.** From (3.18) we can derive the following equivalent formulas for $\lambda_n$

$$\begin{aligned}\lambda_n &= \frac{\beta_{n+2}\,\Delta\beta_n\,\Delta\tilde{S}_{n+2} - \beta_n\,\Delta\beta_{n+1}\,\Delta\tilde{S}_{n+1}}{\Delta\beta_n\,\Delta\tilde{S}_{n+1} - \Delta\beta_{n+1}\,\Delta\tilde{S}_n} \\ &= \frac{\beta_{n+2}\dfrac{\Delta\tilde{S}_{n+2}}{\Delta\tilde{S}_{n+1}} - \beta_n\dfrac{\Delta\beta_{n+1}}{\Delta\beta_n}}{1 - \dfrac{\Delta\beta_{n+1}}{\Delta\beta_n}\dfrac{\Delta\tilde{S}_n}{\Delta\tilde{S}_{n+1}}}\,.\end{aligned}$$

where
$$\frac{\Delta\beta_{n+1}}{\Delta\beta_n} = \frac{a_{n+1}}{a_{n+3}}\left(\frac{a_{n+2}^2 - a_{n+1}\,a_{n+3}}{a_{n+1}^2 - a_n\,a_{n+2}}\right) = \beta_{n+1}\,\beta_{n+2}\,\gamma_n.$$

Making use of Lemmas 3.1 and 3.2, we obtain

$$\begin{aligned}\lim_{n\to\infty}\lambda_n &= \lim_{n\to\infty}\frac{\beta_{n+2}\dfrac{\Delta\tilde{S}_{n+2}}{\Delta\tilde{S}_{n+1}} - \beta_n\beta_{n+1}\beta_{n+2}\gamma_n}{1 - \beta_{n+1}\beta_{n+2}\gamma_n\dfrac{\Delta\tilde{S}_n}{\Delta\tilde{S}_{n+1}}} \\ &= \lim_{n\to\infty}\frac{\beta_{n+2}\dfrac{\Delta\tilde{S}_{n+2}}{\Delta\tilde{S}_{n+1}} - \beta_n\beta_{n+1}\beta_{n+2}\gamma_n}{1 - \beta_{n+1}^2\beta_{n+2}\gamma_n\left(\beta_{n+1}\dfrac{\Delta\tilde{S}_{n+1}}{\Delta\tilde{S}_n}\right)^{-1}} \\ &= \frac{\lambda - \beta^3\gamma}{1 - \dfrac{\beta^3\gamma}{\lambda}} = \lambda\frac{\lambda - \beta^3\gamma}{\lambda - \beta^3\gamma} = \lambda.\end{aligned}$$

□

We stress that $\beta_n$ and $\gamma_n$ only depend on the sequence $(a_n)$. Therefore, since in our study $(a_n)$ is assumed to be known, we are able to check if sequences $(\beta_n)$ and $(\gamma_n)$ have a limit and predict for which $\lambda$ the estimate might not converge to the correct limit.

Note also that, if $a_{n+1}^2 - a_n a_{n+2} = 0$ for some $n$, $\gamma_n$ is not well-defined. Nonetheless, if $a_{n+2}^2 - a_{n+1}a_{n+3} \neq 0$, we can skip this iteration, and compute the next one. If both the numerator and the denominator of $\gamma_n$ are equal to zero, then also the denominator of $\lambda$ as expressed in (3.13) is equal to zero. Hence, if $\gamma_n$ is not well-defined, $\lambda_n$ cannot be computed.

Let us now study the convergence and acceleration properties of transformation ${}_2T_n$. We consider two cases; the sequence $\tilde{S}_n$ converges to $S$ or diverges.

- If the sequence $\tilde{S}_n$ is convergent, then from the third expression of formula (3.11) we have

$$ {}_2T_n = \frac{a_{n+1}\tilde{S}_n\lambda_n - a_n\tilde{S}_{n+1}}{a_{n+1}\lambda_n - a_n} = \frac{\tilde{S}_n\lambda_n - \frac{a_n}{a_{n+1}}\tilde{S}_{n+1}}{\lambda_n - \frac{a_n}{a_{n+1}}}, $$

thus

$$ {}_2T_n - S = \frac{(\tilde{S}_n - S)\lambda_n - \beta_n(\tilde{S}_{n+1} - S)}{\lambda_n - \beta_n}. \tag{3.19} $$

From this expression, it is easy to prove the following theorem on the convergence of transformation ${}_2T_n$.

**Theorem 3.2.** *Transformation ${}_2T_n$ converges to $S$ under the following conditions:*

1. $\lim_{n\to\infty} \tilde{S}_n = S$ ,
2. *there exist $N \in \mathbb{N}$ and $\delta > 0$ such that $|\lambda_n - \beta_n| > \delta$ for every $n > N$.*

**Proof.** By using the hypotheses 1 and 2, (3.19) results in ${}_2T_n - S \to 0$, thus ${}_2T_n \to S$.

□

In order to prove that transformation ${}_2T_n$ accelerates the convergence of sequences of the form (3.17) we will need the following lemma.

**Lemma 3.3.** *If $\dfrac{g_n}{a_n\lambda^n} \to 0$, then*

$$ \lim_{n\to\infty} \beta_n \frac{\tilde{S}_{n+1} - S}{\tilde{S}_n - S} = \lambda\,. $$

**Proof.** We have the following equivalent expressions

$$\begin{aligned}\lim_{n\to\infty}\beta_n\frac{\tilde{S}_{n+1}-S}{\tilde{S}_n-S} &= \lim_{n\to\infty}\beta_n\frac{a_{n+1}\,\lambda^{n+1}+g_{n+1}}{a_n\,\lambda^n+g_n} = \lim_{n\to\infty}\beta_n\frac{a_{n+1}}{a_n}\lambda\frac{1+\dfrac{g_{n+1}}{a_{n+1}\,\lambda^{n+1}}}{1+\dfrac{g_n}{a_n\,\lambda^n}}\\ &= \lim_{n\to\infty}\lambda\frac{1+\dfrac{g_{n+1}}{a_{n+1}\,\lambda^{n+1}}}{1+\dfrac{g_n}{a_n\,\lambda^n}} = \lambda.\end{aligned}$$

□

**Theorem 3.3.** *Under the assumptions of Theorem 3.1, transformation ${}_2T_n$ accelerates the convergence of the sequence (3.17).*

**Proof.** From (3.19) we obtain

$$\frac{{}_2T_n-S}{\tilde{S}_n-S}=\frac{\lambda_n-\beta_n\frac{\tilde{S}_{n+1}-S}{\tilde{S}_n-S}}{\lambda_n-\beta_n}.$$

From Theorem 3.1 it holds that $\lambda_n\to\lambda$ and using the result of Lemma 3.3, the numerator converges to zero. Thus, under the assumption that $\beta\neq\lambda$, we conclude that

$$\lim_{n\to\infty}\frac{{}_2T_n-S}{\tilde{S}_n-S}=0.$$

□

Notice that this kind of result can be given for any estimate $\tilde{\lambda}_n$ that converges to $\lambda$. Moreover, the convergence of $\lambda_n$ to $\lambda$ is the key to prove acceleration. Indeed, it is not necessary to prove the convergence of ${}_2T_n$ to $S$.

- Let $\tilde{S}_n$ be a divergent sequence. Then from the equation (3.19), we have

$$\begin{aligned}{}_2T_n-S &= \frac{\lambda_n(a_n\lambda^n+g_n)-\beta_n(a_{n+1}\lambda^{n+1}+g_{n+1})}{\lambda_n-\beta_n}\\ &= \frac{a_n\lambda^n(\lambda_n-\lambda)+\lambda_n g_n-\beta_n g_{n+1}}{\lambda_n-\beta_n}\\ &= \frac{a_n\lambda^n(\lambda_n-\lambda)}{\lambda_n-\beta_n}+\frac{\lambda_n g_n-\beta_n g_{n+1}}{\lambda_n-\beta_n}.\end{aligned}$$

Note that it is easy to give a condition so that $(\lambda_n g_n - \beta_n g_{n+1})/(\lambda_n - \beta_n)$ converges to zero. However, since $a_n\lambda^n$ is now a divergent sequence, it is not simple to find assumptions under which $(a_n\lambda^n(\lambda_n - \lambda))/(\lambda_n - \beta_n)$ does not diverge. In particular, under the hypotheses of Theorem 3.1, we have $\lambda_n - \lambda \to 0$, but we did not find any meaningful condition that ensures the convergence of ${}_2T_n$. Numerical experiments show that transformation ${}_2T_n$ does not converge when the sequence diverges.

**Remark 3.4.** *If $\lambda_n$ rapidly converges to $\lambda$ and $a_n\lambda^n$ does not diverge too quickly at the beginning, ${}_2T_n$ may have a semi-convergent behavior, that is it converges during the first iterations, but then it diverges (for the concept of semi-convergence see [64] and also [72, Appendix E]).*

**Remark 3.5.** *If $\lambda_m = \lambda$ for a certain $m$, then $T_m = S + \varepsilon$, where the quantity $\varepsilon = (\lambda_m g_m - \beta_m g_{m+1})/(\lambda_m - \beta_m)$ can be supposed to be very small (converging to zero). This may explain why for diverging sequences we sometimes have values of ${}_2T_n$ which are very close to $S$, even if the transformation generally diverges.*

## 3.3 Numerical experiments

In this Section, we test the convergence of the different approximations of $\lambda$ and we also compare our best transformation, namely ${}_2T_n$, with some well-known transformations and the transformations presented in [19]. As an application, we evaluate the digamma function using different transformations. Finally, we test transformation ${}_2T_n$ in some special cases in which the convergence of $\lambda_n$ to $\lambda$ is not ensured thus the transformation could fail.

All the experiments were performed using `Matlab 7.12.0`. In some cases, when computing $\lambda$ or $T_n$ by solving a linear system, a singular matrix appears. Thus, when a circle $\circ$ appears in a figure, it denotes the corresponding iteration. We also use the symbol $\times$ to indicate the points where $T_n$ or $\lambda$ are computed at machine precision, thus the error is zero. Whenever we computed $\lambda$ as the solution of systems (3.4) or (3.16), we used the `Matlab` backslash command \.

Note that all the experiments were performed using as ${}_1T_n$ the expression (3.9) with denominator

$$D_n = a_{n+3}S_{n+2}(a_{n+1}^2 - a_n a_{n+2}) + a_{n+2}S_{n+1}(a_n a_{n+3} - a_{n+1}a_{n+2}) + a_{n+1}S_n(a_{n+2}^2 - a_{n+1}a_{n+3}).$$

For both ${}_2T_n$, ${}_3T_n$ we use the third expression of (3.12), with $\lambda$ computed as solution of the system (3.4) and (3.16), respectively.

### 3.3.1 Testing the proposed transformations

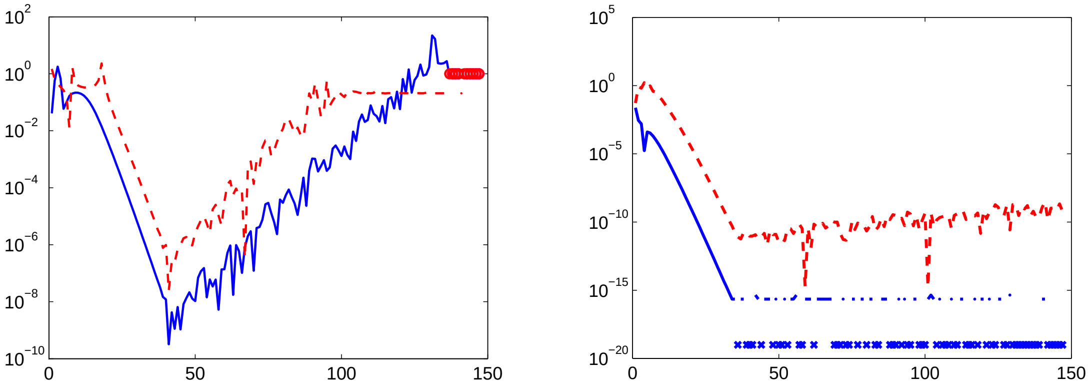
In order to test our transformations we consider two sequences. The first one is convergent and satisfies the condition under which ${}_2T_n$ accelerates it (Theorem 3.3). The second one is divergent and such that ${}_2T_n$ is expected to semi-converge, according to the theoretical analysis developed in Section 3.2. First, we introduce the sequence

$$S_n = 1 + \log\left(1 + \frac{1}{n}\right)\left(\frac{4}{5}\right)^n + e^{-n}(1 + n^2) \tag{3.20}$$

with convergence rate $\rho = \frac{4}{5}$, thus $S_n$ converges linearly, and $\beta = \lim_{n\to\infty} a_n/a_{n+1} = 1$. Then, we consider the sequence

$$S_n = 1 + \left(10\sin\left(\pi\left(1 + \frac{1}{n^2}\right)\right) + 2\cos\left(\pi\left(1 + \frac{1}{n^2}\right)\right)\right)\left(-\frac{6}{5}\right)^n + e^{-n}(1 + n^2) \tag{3.21}$$

which is alternating, divergent and has $\beta = \lim_{n\to\infty} a_n/a_{n+1} = 1$.



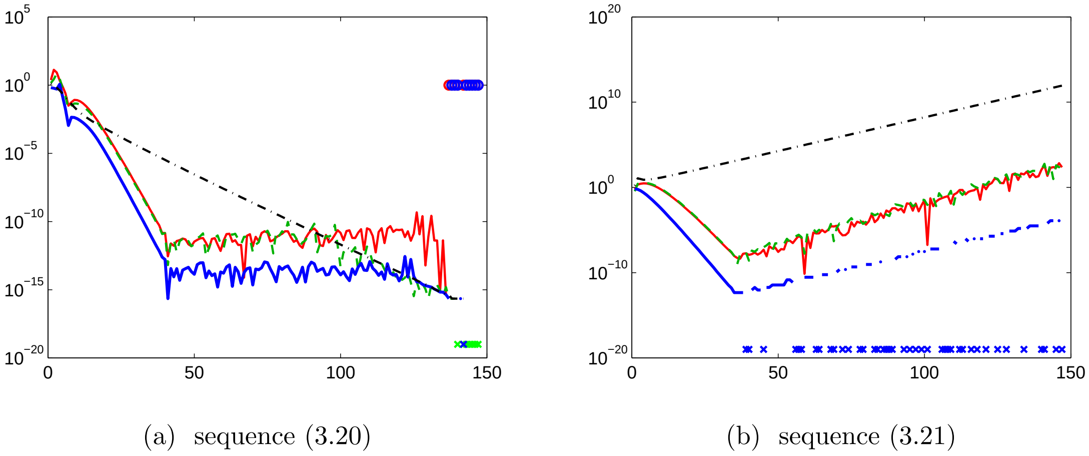
(a) sequence (3.20)

(b) sequence (3.21)

Figure 3.1: Comparison of the absolute value of the error in the estimate of $\lambda$ solving systems (3.4) (solid line), and (3.16) (dashed line).

Figure 3.1 illustrates the absolute error of the estimate of $\lambda$ obtained as a solution of the system (3.4) or (3.16). We notice that for both the testing sequences the first system gives a better approximation than the second one. In particular, for the sequence (3.21) we reach machine precision after a few iterations (see Figure 3.1b). A different case appears in Figure 3.1a, where both approximations reach a good precision before diverging. This semi-convergence is due to rounding errors that appear in the solution of the systems (3.4) and (3.16) since, for converging sequences, $\Delta S_n \to 0$.

Let us now compare the performance of transformations ${}_1T_n$, ${}_2T_n$, ${}_3T_n$. We stress that, when dealing with convergent sequences, we consider as the best transformation the one that converges to $S$ in a few iterations and with good precision. Instead, for diverging sequences, semi-convergence is what we expect from a good transformation. In Figure 3.2 the absolute errors of transformations ${}_1T_n$, ${}_2T_n$, ${}_3T_n$ are illustrated. We observe that all the transformations accelerate the convergence of the sequence even in the case of the sequence (3.20) where we had a difficulty estimating $\lambda$ (see figures 3.1a and 3.2a). In all the examples, the best result is obtained from transformation ${}_2T_n$. Note that for every $n$ all the transformations use the same sequence of terms, that is $S_n, S_{n+1}, S_{n+2}, S_{n+3}$.



(a) sequence (3.20) (b) sequence (3.21)

Figure 3.2: Comparison of $|S - S_n|$ (dash-dotted line) and $|S - T_n|$ for transformations ${}_1T_n$ (dashed line), ${}_2T_n$ (bold solid line) and ${}_3T_n$ (solid line).

### 3.3.2 Comparison with other transformations

We now compare transformation ${}_2T_n$ with several well-known transformations, and the three transformations introduced by Brezinski and Redivo-Zaglia in [19].

In Figure 3.3, we plot $|S - T_n|$, where $T_n$ is one of the following transformations:

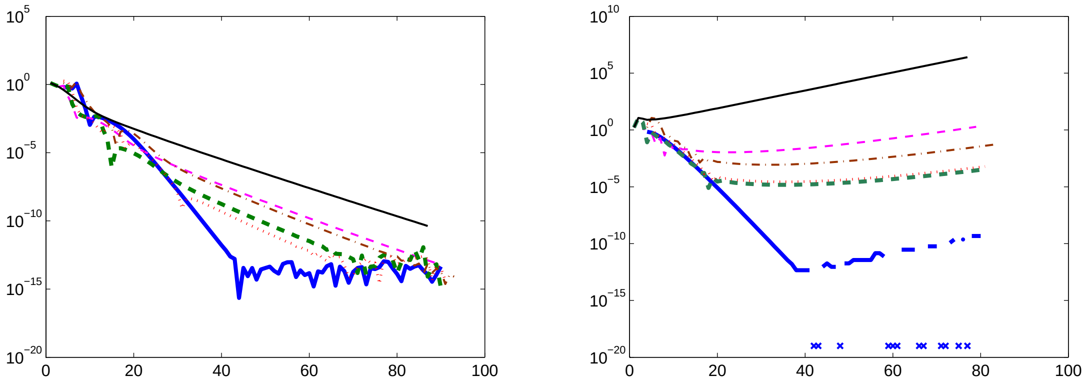
- transformation ${}_2T_n$, which uses four terms of the sequence;

- $\varepsilon$-algorithm ($\varepsilon_{2k}^{(n)}$) with $k = 2$ (cf. Section 1.4), using five terms;

- Aitken's $\Delta^2$ process, implemented with the $\varepsilon$-algorithm, since $T_n = \varepsilon_2^{(n)}$ , where $T_n$ is the Aitken's $\Delta^2$ process (see [69, Eq. (5.1-5)]). This transformation uses three terms.

- algorithm $\theta_2^{(n)}$ (see [18] and [69]), which uses four terms;

- Levin type transformation $\mathcal{L}_k^{(n)}(\beta, S_n, \omega_n)$ (see [18], [37] and [49]). For $\omega_n$ we use the formula that gives the $u$-transformation and we set $k = 3$ so that the transformation uses four terms. The parameter $\beta$ is chosen equal to 1, that is the optimal choice for our sequences following the procedure described in [1]. However, other values of $\beta$ gave similar results in our experiments.



(a) sequence (3.20)

(b) sequence (3.21)

Figure 3.3: Comparison of $|S - S_n|$ (solid line) and $|S - T_n|$ values using Aitken's $\Delta^2$ process (dashed line), $\varepsilon_4^{(n)}$ (bold dashed line), $\theta_2^{(n)}$ (dash-dotted line), Levin type transformation (dotted line) and transformation ${}_2T_n$ (bold solid line).

Figure 3.3a shows clearly that, for the sequence (3.20), transformation ${}_2T_n$ converges faster than the other transformations. Considering the divergent sequence (3.21), all the transformations semi-converge but ${}_2T_n$ reaches a higher accuracy before diverging (see Figure 3.3b). The good performance of ${}_2T_n$ comes as no surprise, since this transformation was built from the kernel (3.1), thus it is expected to perform well for sequences of the type of Eq. (3.17).

In the sequel we compare transformation ${}_2T_n$ with the transformations proposed in [19]. In that paper, Brezinski and Redivo-Zaglia considered the kernel consisting of sequences of the form

$$\begin{array}{rcll} S_n & = & S+(a+bx_n)\lambda^n, & n=0,1,\ldots, \\ \text{or} \quad S_n & = & S+(a+bx_n)^{-1}\lambda^n, & n=0,1,\ldots, \end{array}$$

where $S, a, b$ and $\lambda$ are unknown numbers and $(x_n)$ a known sequence. These kernels obviously contain (1.2).

For the first class of sequences two transformations are proposed, namely

$$ {}_4T_n = S_{n+1} - \frac{\Delta S_{n+1} - \lambda^2 r_n \Delta S_n}{(\lambda r_n - 1)(1-\lambda)}, \text{ with } r_n = \frac{\Delta x_{n+1}}{\Delta x_n}, \tag{3.22}$$

and

$$ {}_5T_n = S_{n+1} + \frac{\Delta S_{n+1} - \lambda^2 \Delta S_n}{(1-\lambda)^2}, \tag{3.23}$$

where the parameter $\lambda$ is evaluated as the solution of two different linear systems.

Sequences of the second form are instead treated with the transformation

$$ {}_6T_n = \frac{N_n}{D_n}, \tag{3.24}$$

where

$$\begin{aligned} N_n &= \lambda^2 S_{n+1}(S_{n+2}-S_n) - 2\lambda(S_{n+3}S_{n+1} - S_{n+2}S_n) + S_{n+2}(S_{n+3}-S_{n+1}), \\ D_n &= \lambda^2(S_{n+2}-S_n) - 2\lambda(S_{n+3}-S_{n+2}+S_{n+1}-S_n) + (S_{n+3}-S_{n+1}). \end{aligned}$$

The unknowns $\lambda$ and $\lambda^2$ are computed by solving the following system as a linear one with unknowns $\lambda, \lambda^2, \lambda S, \lambda^2 S, S$

$$\begin{aligned} &\lambda^2 S_{n+1+i}(S_{n+2+i}-S_{n+i}) - 2\lambda(S_{n+1+i}S_{n+3+i} - S_{n+i}S_{n+2+i}) - \lambda^2(S_{n+2+i}-S_{n+i}) \\ &+2\lambda S(S_{n+3+i}-S_{n+2+i}+S_{n+1+i}-S_{n+i}) - S(S_{n+3+i}-S_{n+1+i}) = \\ &-S_{n+2+i}(S_{n+3+i}-S_{n+1+i}), \quad i=0,\ldots,4. \end{aligned}$$

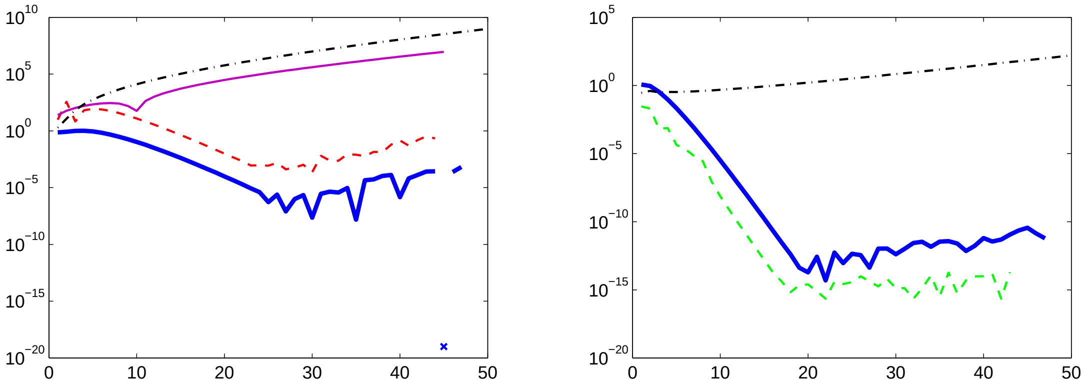
We will test the transformations on the same sequences used in [19], namely

$$S_n = S + \lambda^n (2 - n^{\frac{7}{2}}) + e^{-n}(1 + n^2), \quad \text{with } \lambda = \frac{23}{20}\,, \tag{3.25}$$

and

$$S_n = S + \lambda^n \frac{1}{\left(2 + \frac{11}{10}n\right)} + \left(\frac{1}{10}\right)^n n^{\frac{5}{2}}, \quad \text{with } \lambda = -\frac{6}{5}\,. \tag{3.26}$$

We would like to stress that for our transformation we took as known the sequence $a_n = a + bx_n$ or $a_n = (a + bx_n)^{-1}$, whereas in [19] $a$ and $b$ are unknowns and only $x_n$ is known. This may explain the good performance of ${}_2T_n$. Indeed, for the sequence (3.25) transformation ${}_2T_n$ converges faster than transformation ${}_4T_n$ and transformation ${}_5T_n$ (see Figure 3.4a). In Figure 3.4b we compare the absolute errors of transformation ${}_6T_n$ and transformation ${}_2T_n$ for the sequence (3.26). We observe that ${}_6T_n$ performs slightly better than ${}_2T_n$. Note that for the implementation of transformation ${}_2T_n$ we used 4 terms of the sequence, while transformations ${}_4T_n$ and ${}_5T_n$ use 6 and 5 terms, correspondingly. For the transformation ${}_6T_n$ we used 8 terms.



(a)

(b)

Figure 3.4: (a) Comparison of $|S - S_n|$ (dash-dotted line) and $|S - T_n|$ using the transformations ${}_4T_n$ (dashed line), ${}_5T_n$ (solid line) and transformation ${}_2T_n$ (bold solid line) on sequence (3.25). (b) Comparison of $|S - S_n|$ (dash-dotted line) and $|S - T_n|$ using transformations ${}_6T_n$ (bold dashed line) and transformation ${}_2T_n$ (bold solid line) on sequence (3.26).

### 3.3.3 Evaluation of the digamma function

We will now see how transformation ${}_2T_n$ works in a summation problem. We consider the following series expansion for the psi or digamma function [2]

$$\begin{aligned} \psi(1+z) &= -\gamma + z\,\mathcal{Z}(z) && (3.27)\\ \mathcal{Z}(z) &= \sum_{\nu=0}^{\infty} \zeta(\nu+2)(-z)^{\nu} && (3.28) \end{aligned}$$

with $\gamma$ Euler's constant and $\zeta(\nu{+}2)$ the Riemann zeta function (see [2] equations (6.1.3) and (23.2.1) respectively or [70] equation 1.2). The infinite series in (3.28) converges for $|z| < 1$.

We rewrite (3.28) as in [70, Eq. 2.1]

$$\begin{aligned} \mathcal{Z}(z) &= \mathcal{Z}_n(z) + \mathcal{R}_n(z), && (3.29)\\ \mathcal{Z}_n(z) &= \sum_{\nu=0}^{n} \zeta(\nu+2)(-z)^{\nu}, && (3.30)\\ \mathcal{R}_n(z) &= (-z)^{n+1} \sum_{\nu=0}^{\infty} \zeta(n+\nu+3)(-z)^{\nu}. && (3.31) \end{aligned}$$

The convergence of the sequence $\mathcal{Z}(z)$ can only be improved if the truncation errors $\mathcal{R}_n(z)$ are transformed into others with better numerical properties.

We replace the zeta functions $\zeta(n+\nu+3)$ in (3.31) by their Dirichlet series and interchange the order of summations, thus we obtain (see [70, Eq. 2.2])

$$\mathcal{Z}_n(z) = \mathcal{Z}(z) - (-1)^{n+1} \sum_{m=0}^{\infty} \frac{[z/(m+1)]^{n+1}}{(m+1)(m+z+1)}. \qquad (3.32)$$

We observe that the partial sums $\mathcal{Z}_n(z)$ are a special case of the following class of sequences with $q_j = z/j$ and $c_j = -1/[j(j+z)]$ (see [70, Eq. 2.3])

$$s_n = s + (-1)^{n+1} \sum_{j=1}^{\infty} c_j (q_j)^{n+1}.$$

Following the treatment presented in [70] by Weniger, we assume that all the $q_j$'s have the same sign and are ordered in magnitude according to

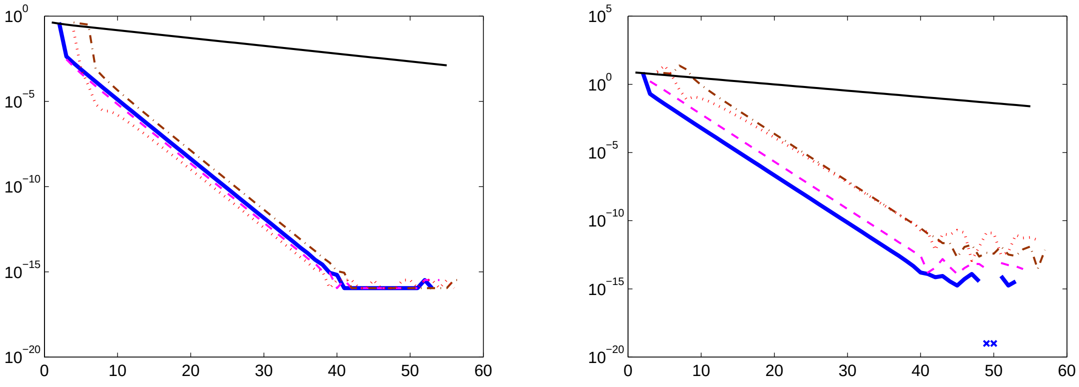
$$1 > |q_1| > |q_2| > ... > |q_\ell| > |q_{\ell+1}| > ... \geq 0,$$

whilst the $c_j$'s are unspecified coefficients.

Note that (3.32) is of the type of Eq. (3.17) with

$$
\begin{aligned}
\tilde{S}_n &= \mathcal{Z}_n(z), \\
S &= \mathcal{Z}(z), \\
a_n &= (-1)^n \frac{z}{z+1}, \\
\lambda &= z, \\
g_n &= (-1)^n \sum_{m=1}^{\infty} \frac{[z/(m+1)]^{n+1}}{(m+1)(m+z+1)}.
\end{aligned}
$$

Since $\lambda$ is already known, we do not need to approximate it for the transformation ${}_2T_n$. Therefore, in order to compute ${}_2T_n$, we only need two terms of the sequence. For this reason, in the numerical experiments we have not considered the algorithm $\varepsilon_4^{(n)}$ for evaluating the digamma function. The performance of the other transformations is illustrated in Figure 3.5. We observe that generally transformation ${}_2T_n$ behaves in the same way or better than Aitken's $\Delta^2$ process, $\theta_2^{(n)}$ algorithm and $u$-transformation. In particular, for negative values of $z$, transformation ${}_2T_n$ reaches the best precision (see Figure 3.5b).



(a) z = 0.9

(b) z = -0.9

Figure 3.5: Digamma function. Comparison of $|\mathcal{Z}(z) - \mathcal{Z}_n(z)|$ (solid line) and $|S - T_n|$ values using Aitken's $\Delta^2$ process (dashed line), $\theta_2^{(n)}$ (dash-dotted line), Levin type transformation (dotted line) and transformation ${}_2T_n$ (bold solid line).

### 3.3.4 Problematic cases

In this section we consider several sequences each of which does not satisfy at least one of the hypotheses of Theorem 3.1. We set $S = 1$, so the sequences have the form

$$\tilde{S}_n = 1 + a_n\lambda^n + g_n, \tag{3.33}$$

where $g_n = (1+n^2)e^{-n}$ is a sequence converging to zero and subdominant to $a_n\lambda^n$.

In figures 3.6, 3.7 and 3.8 on the left we plot the absolute error of the estimate of $\lambda$ by solving the system (3.4), using the `Matlab` backslash operator \. On the right, we compare the absolute error of the transformations ${}_2T_n$, Aitken's $\Delta^2$ process, $\varepsilon$-algorithm, $\theta_2^{(n)}$ algorithm and $u$-transformation.

In the first two examples we assume that $\beta = \lambda$, thus the sequences converge logarithmically. We already know that Aitken's $\Delta^2$ process and $\varepsilon$-algorithm are not able to accelerate such a sequence (see for example [18]). So, we test ${}_2T_n$ on a logarithmically convergent sequence in order to understand if it fails in accelerating the sequence as Aitken's $\Delta^2$ process and Wynn's $\varepsilon$-algorithm do.

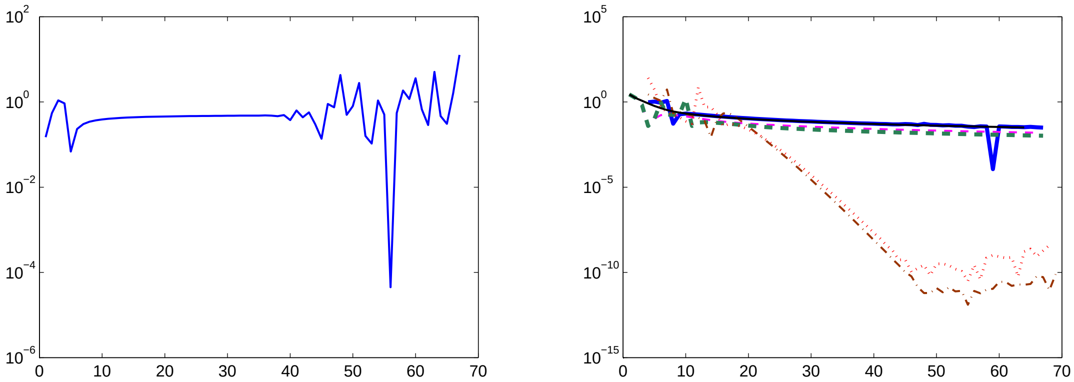


Figure 3.6: On the left, the errors in the estimate of $\lambda$ (obtained by system (3.4)) for sequence (3.34). On the right, the values $|S - S_n|$ (solid line) are compared with the errors obtained using Aitken's $\Delta^2$ process (dashed line), $\varepsilon_4^{(n)}$ (bold dashed line), $\theta_2^{(n)}$ (dash-dotted line), Levin type transformation (dotted line) and transformation ${}_2T_n$ (bold line).

To start with, we consider (3.33) with $a_n = \dfrac{\left(\frac{5}{4}\right)^{n+\frac{16}{5}}}{n}$ and $\lambda = \frac{4}{5}$, thus we obtain the sequence

$$S_n = 1 + \frac{1}{n}\left(\frac{5}{4}\right)^{n+\frac{16}{5}}\left(\frac{4}{5}\right)^n + (1+n^2)e^{-n}. \tag{3.34}$$

We see that $\beta = \lambda$ and $\gamma = \beta^{-2}$. Therefore the last condition of Theorem 3.1 is violated. This explains the curve in Figure 3.6a, which shows that $\lambda_n$ fails to converge to $\lambda$. The transformation ${}_2T_n$ seems to be affected, too; it converges to $S$ but it fails to accelerate the convergence of the sequence, like Aitken's $\Delta^2$ process and $\varepsilon$-algorithm. Instead, $\theta_2^{(n)}$ and $u$-transformation succeed a good acceleration.

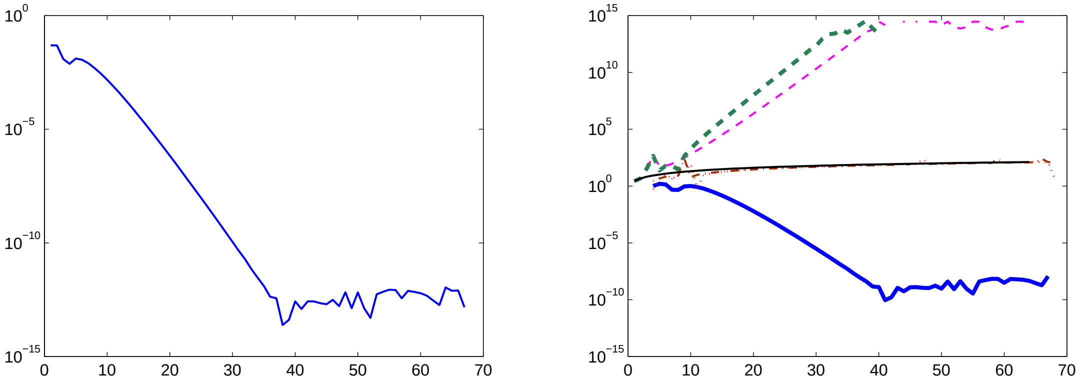


Figure 3.7: On the left, the errors in the estimate of $\lambda$ (obtained by system (3.4)) for sequence (3.35). On the right, the values $|S - S_n|$ (solid line) are compared with the errors obtained using Aitken's $\Delta^2$ process (dashed line), $\varepsilon_4^{(n)}$ (bold dashed line), $\theta_2^{(n)}$ (dash-dotted line), Levin type transformation (dotted line) and transformation ${}_2T_n$ (bold solid line).

Let us now consider a diverging sequence $\tilde{S}_n$ such that $\lambda = \beta$. We stress that $\tilde{S}_n$ cannot be a sequence with alternating sign, since in this case we can rewrite it as $\tilde{S}_n = (-1)^n a_n \lambda^n + g_n$, with $a_n$ and $\lambda$ positive for every $n$. Then, $\beta = \lim_{n\to\infty} -\dfrac{a_n}{a_{n+1}} < 0$, while $\lambda > 0$. Therefore, $\tilde{S}_n$ has to be a sequence with positive terms. Note that the summation of this kind of sequences can be very difficult (for more details see for example [71, pp. 15-17]).

Setting $a_n = n\left(\frac{5}{4}\right)^{n+\frac{16}{5}}$, we obtain the following divergent sequence

$$S_n = 1 + n\left(\frac{5}{4}\right)^{n+\frac{16}{5}}\left(\frac{4}{5}\right)^n + (1+n^2)e^{-n}. \tag{3.35}$$

Once more, $\beta = \lambda$ and $\gamma = \beta^{-2}$, that is the fourth hypothesis of Theorem 3.1 is not satisfied. Nevertheless, unlike the previous example, transformation ${}_2T_n$ performs well and $\lambda_n$ converges to $\lambda$ (see Figure 3.7). Therefore we reach the conclusion that our assumption is sufficient but not necessary. As for the other transformations, Aitken's process and $\varepsilon$-algorithm diverge, $\theta_2^{(n)}$ and $u$-transformation diverge at the same rate of the sequence. We note that the semi-convergence of ${}_2T_n$ agrees with Remark 3.4.

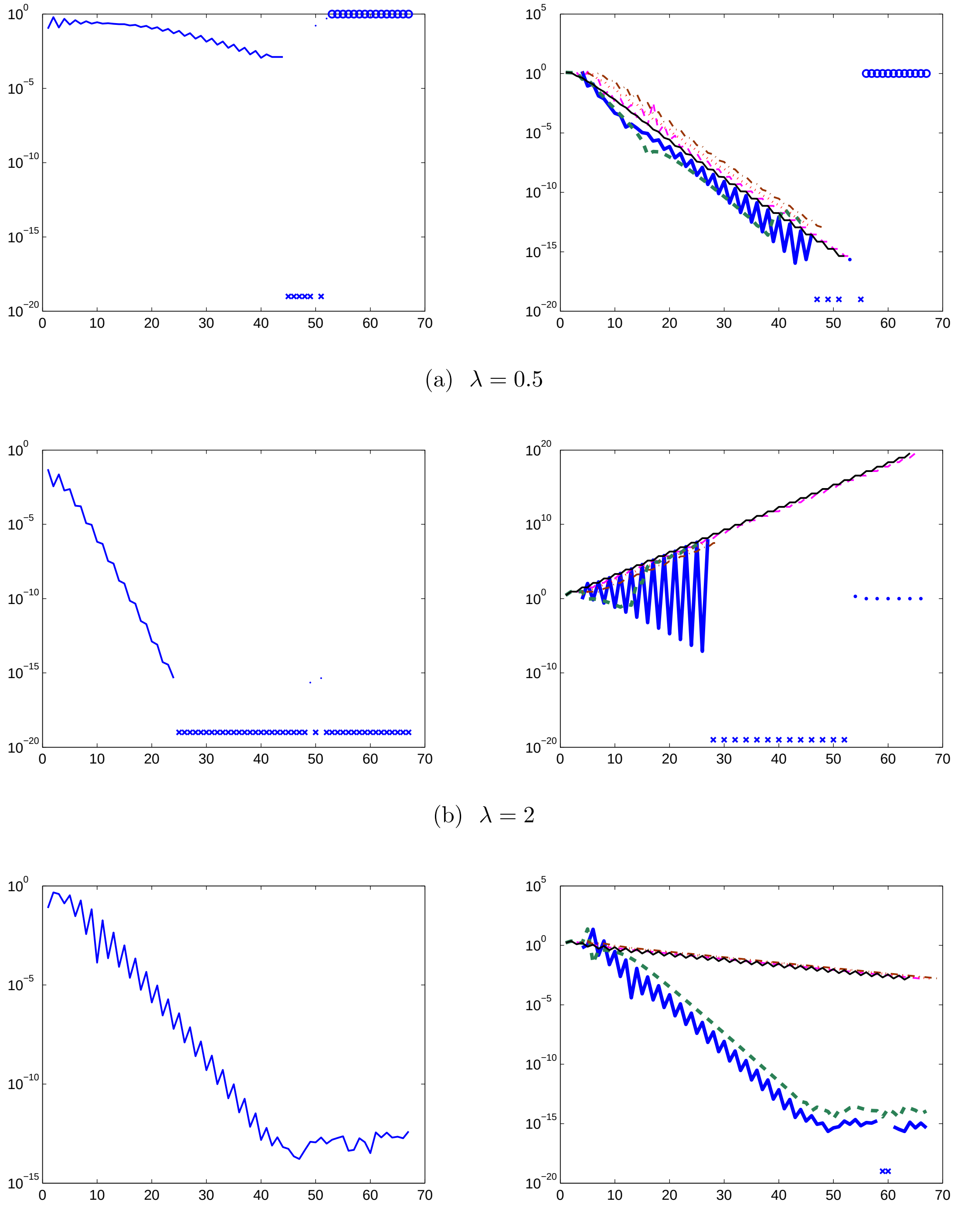
**Remark 3.6.** *We recall that one of the conditions under which transformation* ${}_2T_n$ *accelerates a sequence* $\tilde{S}_n$ *is* $\beta \neq \lambda$ *(see Theorem 3.3). Moreover, we have seen that if* $\tilde{S}_n$ *is convergent, then* $\beta = \lambda$ *if and only if* $\tilde{S}_n$ *converges logarithmically. In addition, sequence* (3.34) *is an example of a logarithmically convegent sequence that* ${}_2T_n$ *is not able to accelerate. We would like to stress that these facts are consistent with the well-known result of Delahaye and Germain-Bonne [31, 32] that there is no sequence transformation that could accelerate the convergence of all logarithmically convergent sequences. Furthermore, we have shown that all these sequences are definitely positive. Hence if we deal with divergent sequences,* $\lambda = \beta$ *means that we sum a monotone sequence. Therefore, even in this case condition* $\beta = \lambda$ *leads to a class of sequences difficult to treat.*

Next, we consider the sequence $a_n = \frac{3}{2} + \frac{(-1)^n}{2}$, which alternatively assumes the values 1 and 2. Using it in (3.33) gives the following sequence

$$S_n = 1 + \left(\frac{3}{2} + \frac{(-1)^n}{2}\right)\lambda^n + (1+n^2)e^{-n}. \tag{3.36}$$

It is easy to verify that $\beta_n$ is a bounded sequence with no limit, taking alternatively the values 2 and 1/2, while $\gamma = -1$. We consider three different cases;

- $\lambda = \frac{1}{2}$: $S_n$ is convergent, $\lambda = \liminf_{n\to\infty}\beta_n$ hence an accumulation point;
- $\lambda = 2$: $S_n$ is divergent, $\lambda = \limsup_{n\to\infty}\beta_n$ hence an accumulation point;
- $\lambda = \frac{9}{10}$: $S_n$ is convergent, $|\lambda - \beta_n| > \frac{1}{2}$ for any $n$ hence ${}_2T_n \to S$ (Theorem 3.2).



(a) $\lambda = 0.5$

(b) $\lambda = 2$

(c) $\lambda = 0.9$

Figure 3.8: On the left, the errors in the estimate of $\lambda$ (obtained by system (3.4)) for sequence (3.36) with different values of $\lambda$. On the right, the values $|S - S_n|$ (solid line) are compared with the errors obtained using Aitken's $\Delta^2$ process (dashed line), $\varepsilon_4^{(n)}$ (bold dashed line), $\theta_2^{(n)}$ (dash-dotted line), Levin type transformation (dotted line) and transformation ${}_2T_n$ (bold line).

According to the results illustrated in Figure 3.8, it seems that the lack of a limit for $\beta_n$ does not affect the convergence. In fact, we get very good results in all three cases. So we conclude that the second condition of Theorem 3.1 is not a necessary condition.

We remark that, when $\lambda = \frac{1}{2}$, the determinant of the system (3.4) is equal to $-3(S_{n+2} - S_n) - 2(S_{n+1} - S_n)$. Therefore, if $\Delta S_n$ is close to zero at machine precision, then singularity problems arise (see Figure 3.8a). However, this is not a real problem, since the reason is that $S_n$ has reached the value of $S$ at machine precision.

In Figure 3.8b we observe that, when $\lambda_n = 2$ at machine precision, the values of transformation ${}_2T_n$ are not computed for $n$ odd. This happens because the denominator in transformation ${}_2T_n$ is alternatively equal to 0 and 1.

Concerning the other transformations, we see that they do not accelerate the convergence of the sequence (3.36) for any of the considered values for $\lambda$, except for $\varepsilon_4^{(n)}$ which semi-converges in case $\lambda = 0.9$. Though, we recall that for the implementation of transformation ${}_2T_n$ we need less terms than for $\varepsilon_4^{(n)}$.

# Chapter 4

# Extrapolation methods for vector sequences

In the rest of this PhD thesis we deal with the acceleration of iterative methods for solving systems of linear equations. In these problems vector sequences arise and we will try to extrapolate them. For this reason, this chapter is devoted to some major vector extrapolation algorithms. More precisely, we revise the vector $\varepsilon$-algorithm obtained by Wynn [74], the topological $\varepsilon$-algorithms explored by Brezinski [13], and the more recent simplified topological $\varepsilon$-algorithms (STEA) proposed by Brezinski and Redivo-Zaglia [21].

## 4.1 The vector $\varepsilon$-algorithm

The vector $\varepsilon$-algorithm, obtained by Wynn [74], is the first algorithm found for accelerating vector sequences. It was obtained directly from the scalar algorithm as a generalization to vectors, thus it shares the same rules (1.17), with the difference that now $\mathbf{S}_n$ is a (real or complex) vector of dimension $N$ and $\varepsilon_{-1}^{(n)}$ is the zero vector. After defining the inverse $\mathbf{y}^{-1}$ of a vector $\mathbf{y}$ as

$$\mathbf{y}^{-1} = \frac{\mathbf{y}}{\|y\|^2},$$

we give the rules for the vector $\varepsilon$-algorithm

$$\begin{cases} \boldsymbol{\varepsilon}_{-1}^{(n)} &= \mathbf{0}, \\ \boldsymbol{\varepsilon}_{0}^{(n)} &= \mathbf{S}_n, \qquad n = 0, 1, ... \\ \boldsymbol{\varepsilon}_{k+1}^{(n)} &= \boldsymbol{\varepsilon}_{k-1}^{(n+1)} + (\boldsymbol{\varepsilon}_{k}^{(n+1)} - \boldsymbol{\varepsilon}_{k}^{(n)})^{-1}, \qquad k, n = 0, 1, ... \end{cases} \tag{4.1}$$

The kernel of the vector $\varepsilon$-algorithm obtained by McLeod [50] in the real case and by Graves-Morris [39] in the complex case is described in the following theorem

**Theorem 4.1.** *If the vector $\varepsilon$-algorithm is applied to a sequence of complex vectors $(\mathbf{S}_n)$ such that $\forall n \geq N$*

$$\sum_{i=0}^{k} a_i(\mathbf{S}_{n+i} - \mathbf{S}) = \mathbf{0} \tag{4.2}$$

*where $\mathbf{S}$ is a vector and $a_i$, $i = 0, \ldots, k$, complex numbers such that $a_k \neq 0$ and $a_0 + \cdots + a_k \neq 0$ then $\forall n \geq N$,*

$$\boldsymbol{\varepsilon}_{2k}^{(n)} = \mathbf{S}.$$

**Remark 4.1.** *Although, the relation* (4.2) *has exactly the same form as the relation defining the kernel of the scalar $\varepsilon$-algorithm* (1.16)*, the condition in Theorem 4.1 is only sufficient while it was also necessary in Theorem 1.1.*

## 4.2 Topological Shanks transformations and topological $\varepsilon$-algorithms

As we saw in the previous section, the vector $\varepsilon$-algorithm was obtained directly from the rule of the scalar algorithm, without a firm theoretical basis. For this reason, Brezinski [13] proposed another way for obtaining a vector generalization of Shanks' transformation and the $\varepsilon$-algorithm. The new generalization, called topological Shanks transformation, is implemented via a recursive algorithm, namely the topological $\varepsilon$-algorithm. This algorithm can be potentially applied to accelerate the convergence of any sequence of elements of a topological vector space $E$, thus the name.

The procedure towards Brezinski's topological $\varepsilon$-algorithm follows the spirit of Shanks for the scalar $\varepsilon$-algorithm. We start from a relation similar to (1.12), i.e.

$$a_0(\mathbf{S}_n - \mathbf{S}) + \cdots + a_k(\mathbf{S}_{n+k} - \mathbf{S}) = \mathbf{0} \in E \tag{4.3}$$

where $\mathbf{S}_n, \mathbf{S} \in E$ and the $\alpha_i$'s are scalars with $a_0\, a_k \neq 0$ and $a_0 + \cdots + a_k \neq 0$. The aim is to compute $\mathbf{S}$ but first we have to compute $\alpha_i$'s. So, we consider the system formed by the equations

$$a_0 \Delta \mathbf{S}_i + \cdots + a_k \Delta \mathbf{S}_{i+k} = \mathbf{0}, \quad i = n, \ldots, n+k-1. \tag{4.4}$$

Now let $\mathbf{y}$ be an arbitrary vector in the algebraic dual space of $E$, $E^*$. We define the bilinear form $< \mathbf{y}, \mathbf{u} >$ as

$$< \mathbf{y}, \mathbf{u} >= \sum_{i=1}^{N} y_i u_i$$

where the $y_i$'s and the $u_i$'s are the components of $\mathbf{y}$ and $\mathbf{u}$ respectively, and we take the duality product of the relations (4.4) with $\mathbf{y}$, that is

$$\alpha_0 < \mathbf{y}, \Delta \mathbf{S}_i > + \cdots + \alpha_k < \mathbf{y}, \Delta \mathbf{S}_{i+k} >= 0, \qquad n, n+1, \ldots, n+k-1. \tag{4.5}$$

We stress that, similarly to the scalar case, if $\mathbf{S}_n$ does not belong to the kernel (4.3), the relations (4.5) still hold, and together with the assumption $a_0 + \cdots + a_k = 1$, they form the following system of $k+1$ equations in $k+1$ unknowns

$$\left\{ \begin{array}{ccccccc} a_0^{(n,k)} & + & \cdots & + & a_k^{(n,k)} & = & 1 \\ a_0^{(n,k)} < \mathbf{y}, \Delta \mathbf{S}_n > & + & \cdots & + & a_k^{(n,k)} < \mathbf{y}, \Delta \mathbf{S}_{n+k} > & = & 0 \\ \vdots & & \vdots & & & & \vdots \\ a_0^{(n,k)} < \mathbf{y}, \Delta \mathbf{S}_{n+k-1} > & + & \cdots & + & a_k^{(n,k)} < \mathbf{y}, \Delta \mathbf{S}_{n+2k-1} > & = & 0 \end{array} \right. . \tag{4.6}$$

The superscript $(n,k)$ denotes the dependence on both $n$ and $k$. From the above system, whose the determinant is assumed to be nonzero, we compute $a_i$'s and then $\mathbf{S}$ can be obtained from

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

for any $i$. When $i = 0$, we have the first topological Shanks transformation, i.e.

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

For $i = k$, the second topological Shanks transformation writes

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

.

*Note*: We will use the notation $\mathbf{e}_k(\mathbf{S}_n)$ every time we want to refer to both or any of the two transformations.

Given the particular form of the right hand side of the system (4.6), topological Shanks transformations can be written as a ratio of determinants, i.e.

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

By construction we have

**Theorem 4.2.** *[13] If $\forall n \geq N$, $\exists a_i$ with $a_0 a_k \neq 0$ and $a_0 + \cdots + a_k \neq 0$ such that*

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

*then $\forall n \geq N$, $\mathbf{e}_k(\mathbf{S}_n) = \mathbf{S}$.*

**Remark 4.2.** *The condition of the last theorem is only sufficient, contrarily to Theorem 1.1, where it was necessary and sufficient.*

For the recursive implementation of the elements $\hat{\mathbf{e}}_k(\mathbf{S}_n), \tilde{\mathbf{e}}_k(\mathbf{S}_n) \in E$ we use the first and second *topological ε-algorithm* (TEA1 and TEA2), respectively, whose common rules are

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

whilst the even rules differ and are given below

$$\text{for TEA1}: \quad \hat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} = \hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} - \hat{\boldsymbol{\varepsilon}}_{2k}^{(n)}}{\langle \hat{\boldsymbol{\varepsilon}}_{2k+1}^{(n+1)} - \hat{\boldsymbol{\varepsilon}}_{2k+1}^{(n)}, \hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} - \hat{\boldsymbol{\varepsilon}}_{2k}^{(n)} \rangle} \in E, \tag{4.7}$$

$$\text{for TEA2}: \quad \tilde{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} = \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}}{\langle \tilde{\boldsymbol{\varepsilon}}_{2k+1}^{(n+1)} - \tilde{\boldsymbol{\varepsilon}}_{2k+1}^{(n)}, \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} \rangle} \in E. \tag{4.8}$$

*Notation:* The inverse of a couple $(\mathbf{u}, \mathbf{y}) \in E \times E^*$ used above is defined as $\mathbf{u}^{-1} = \mathbf{y}/\langle \mathbf{y}, \mathbf{u} \rangle \in E^*$ and $\mathbf{y}^{-1} = \mathbf{u}/\langle \mathbf{y}, \mathbf{u} \rangle \in E$ [13].

**Remark 4.3.** *When the dimension of the vectors is one, the rules of the topological $\varepsilon$-algorithms reduce to the rules of the scalar $\varepsilon$-algorithm.*

An array as that displayed in Table 1.1 for the elements of the scalar $\varepsilon$-algorithm can be formed from the elements of the topological $\varepsilon$-algorithm.

As in the scalar case, the topological $\varepsilon$-algorithm is connected to the topological Shanks transformation via the following relations

$$\boldsymbol{\varepsilon}_{2k}^{(n)} = \mathbf{e}_n(\mathbf{S}_n), \quad \boldsymbol{\varepsilon}_{2k+1}^{(n)} = \mathbf{y}/\langle \mathbf{y}, \mathbf{e}_n(\Delta \mathbf{S}_n) \rangle.$$

The connection between the scalar and the topological $\varepsilon$-algorithms is described below

$$\langle \mathbf{y}, \boldsymbol{\varepsilon}_{2k}^{(n)} \rangle = e_k(\langle \mathbf{y}, \mathbf{S}_n \rangle).$$

*Note*: We stress that in this Ph.D. thesis we always consider $E = \mathbb{R}^N$, thus $E^* = \mathbb{R}^N$ and $\langle \cdot, \cdot \rangle$ is the standard inner product.

## 4.3 Simplified topological $\varepsilon$-algorithms

Recently Brezinski and Redivo-Zaglia [21] introduced two new algorithms for implementing the topological Shanks transformations. These new algorithms called simplified topological $\varepsilon$-algorithms are better than the old topological $\varepsilon$-algorithms both from a theoretical point of view and for numerical reasons.

The starting point for defining the simplified topological $\varepsilon$-algorithm is to apply the scalar $\varepsilon$-algorithm of Wynn to the sequence

$$(S_n) = (\langle \mathbf{y}, \mathbf{S}_n \rangle). \tag{4.9}$$

Then, based on algebraic properties of the topological $\varepsilon$-algorithms and making use of the recursive rule of the scalar $\varepsilon$-algorithm, we obtain the *first simplified topological $\varepsilon$-algorithm* (STEA1) written in one of the following equivalent formulas

$$\begin{aligned}
\hat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{1}{\big(\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}\big)\big(\varepsilon_{2k+1}^{(n+1)} - \varepsilon_{2k+1}^{(n)}\big)}\big(\hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} - \hat{\boldsymbol{\varepsilon}}_{2k}^{(n)}\big)\,, && (4.10)\\
\hat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+1}^{(n)} - \varepsilon_{2k-1}^{(n+1)}}{\varepsilon_{2k+1}^{(n+1)} - \varepsilon_{2k+1}^{(n)}}\big(\hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} - \hat{\boldsymbol{\varepsilon}}_{2k}^{(n)}\big)\,, && (4.11)\\
\hat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}}\big(\hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} - \hat{\boldsymbol{\varepsilon}}_{2k}^{(n)}\big)\,, && (4.12)\\
\hat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \big(\varepsilon_{2k+1}^{(n)} - \varepsilon_{2k-1}^{(n+1)}\big)\big(\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}\big)\big(\hat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} - \hat{\boldsymbol{\varepsilon}}_{2k}^{(n)}\big)\,, && (4.13)
\end{aligned}$$

$k, n = 0, 1, \ldots$, with $\hat{\boldsymbol{\varepsilon}}_0^{(n)} = \mathbf{S}_n \in E$, $n = 0, 1, \ldots$

and the *second simplified topological $\varepsilon$-algorithm* (STEA2) written in one of the following equivalent formulas

$$\begin{aligned}
\tilde{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{1}{\big(\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}\big)\big(\varepsilon_{2k+1}^{(n+1)} - \varepsilon_{2k+1}^{(n)}\big)}\big(\tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}\big)\,, && (4.14)\\
\tilde{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+1}^{(n+1)} - \varepsilon_{2k-1}^{(n+2)}}{\varepsilon_{2k+1}^{(n+1)} - \varepsilon_{2k+1}^{(n)}}\big(\tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}\big)\,, && (4.15)\\
\tilde{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}}\big(\tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}\big)\,, && (4.16)\\
\tilde{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} &= \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \big(\varepsilon_{2k+1}^{(n+1)} - \varepsilon_{2k-1}^{(n+2)}\big)\big(\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}\big)\big(\tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \tilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}\big)\,, && (4.17)
\end{aligned}$$

$k, n = 0, 1, \ldots$, with $\tilde{\boldsymbol{\varepsilon}}_0^{(n)} = \mathbf{S}_n \in E$, $n = 0, 1, \ldots$

For a more detailed description on the derivation of the aforementioned formulas refer to [21].

Several convergence and acceleration results for the simplified topological $\varepsilon$-algorithm are given in [21]. For instance, let us consider the class TM of the totally monotonic sequences, defined as

$$\mathrm{TM} = \big\{(\mathbf{S}_n) \;\text{ such that }\; (-1)^k \Delta^k \mathbf{S}_n \geq 0, \;\; \forall k,\; n, \;\text{ component-wise}\big\}\;,$$

For this class of sequences in [21, Theorem 6.1] the authors give interesting theoretical results. Here we report the one that we will need in Section 6.3.2.

**Theorem 4.3.** *If* $(\mathbf{S}_n)$ *converges to* $\mathbf{S}$, *if*

$$(S_n = \langle \mathbf{y}, \mathbf{S}_n \rangle) \in TM, \tag{4.18}$$

*if exist* $a \neq 0$ *and* $\mathbf{c} \in \mathbb{R}^N$ *such that*

$$(a\mathbf{S}_n + \mathbf{c}) \in TM\,, \tag{4.19}$$

*and if*

$$\lim_{n\to\infty} \frac{\langle \mathbf{y}, \mathbf{S}_{n+1} - \mathbf{S}\rangle}{\langle \mathbf{y}, \mathbf{S}_n - \mathbf{S}\rangle} \neq 1\,, \tag{4.20}$$

*then for all* $k$ *fixed,*

$$\lim_{n\to\infty} \frac{\|\widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n)} - \mathbf{S}\|}{\|\mathbf{x}_{n+2k} - \mathbf{S}\|} = 0\,.$$

From a numerical point of view, the simplified topological $\varepsilon$-algorithm is better than the topological $\varepsilon$-algorithm, since it helps us to overcome several computational problems that characterize the classical topological $\varepsilon$-algorithms. Indeed, computing the scalars present in the ratios in (4.10)-(4.17) instead of performing the difficult manipulations with elements of the dual vector space $E^*$ required in (4.7)-(4.8), reduces the computational cost. Furthermore, we can use the particular rules for the scalar $\varepsilon$-algorithm, in order to handle the potential singularities during the computation of the scalars in (4.10)-(4.17), thus improve the overall numerical stability. Finally, the simplified topological $\varepsilon$-algorithms require the storage of less elements compared to the topological $\varepsilon$-algorithms, since only elements with an even index are used and computed. Indeed, the $\varepsilon$-array corresponding to the simplified topological $\varepsilon$-algorithm is as the one displayed in Table 1.1, but it contains only vectors with even lower indices.

# Chapter 5

# Iterative regularization methods

In this chapter we present several iterative regularization methods that are commonly used for the solution of a linear system of equations. We start with the class of Algebraic Reconstruction Techniques and then we revise Simultaneous Iterative Reconstruction Techniques. We discuss extensively the original method of Cimmino, we generalize it and we propose a new variant. The chapter is concluded with the projected SIRT methods and the semi-iterative methods (also called Accelerated Landweber methods).

## 5.1 Linear inverse ill-posed problems

Inverse problems arise every time we look for the cause of an observed effect. In mathematical terms, we have a known system and the output is also known often with errors. We then have to compute the input. Inverse problems are usually ill-posed in the Hadamard sense, i.e., a solution might not exist, or if it exists it might neither be unique nor continuously depend on the data. When considering numerical methods for inverse problems, we are particularly concerned with the latter issue, since even small perturbations in the data might result in a meaningless solution that is heavily corrupted by the error components. For this reason, some sort of regularization has to be applied. An extensive study on how to treat, analytically and numerically, this kind of problems can be found for instance in [34, 41]. In the next sections we revise several iterative regularization techniques for discretizations of inverse problems.

Let the known system be represented by the matrix $A \in \mathbb{R}^{M \times N}$, the output as the right-hand side $\mathbf{b} \in \mathbb{R}^M$, containing the known data, and the solution be $\mathbf{x} \in \mathbb{R}^N$. Then the problem can be formulated as the following system of linear equations

$$A\mathbf{x} = \mathbf{b}, \tag{5.1}$$

where the matrix $A$ is typically a discretization of an ill-posed problem. The exact solution $\hat{\mathbf{x}} = A^{-1}\mathbf{b}$ is the unique intersection point of the $M$ hyperplanes in $\mathbb{R}^N$

$$\langle \mathbf{a}_i, \mathbf{x} \rangle = b_i,\ \ i = 1, 2, ..., N\,. \tag{5.2}$$

These hyperplanes will play a fundamental role in the definition of the so called row-action methods presented in the next sections. Row action methods can be categorized in two classes, Algebraic Reconstruction Techniques (ART) and Simultaneous Iterative Reconstruction Techniques (SIRT). The representative of each class is Kaczmarz and Cimmino method, respectively, but we will briefly describe all the other methods.

## 5.2 Algebraic Reconstruction Techniques (ART)

Algebraic Reconstruction Techniques (ART) are fully sequential, i.e., each equation is treated at a time, since each equation is dependent on the previous ones. In the following we briefly describe the ART methods. For a deeper understanding of these methods the interested reader may consult [44, 58] and the references therein.

- **Kaczmarz method.** The method is named after the Polish mathematician Stefan Kaczmarz who in 1937 proposed an iterative algorithm for solving linear systems of equations [46]. In 1970 Gordon et al. rediscovered Kaczmarz's method applied in medical imaging; they called it ART [38]. To avoid confusion, we will refer to this method as Kaczmarz method, whilst ART will refer to the entire class of algebraic reconstruction techniques.

  One iteration of Kaczmarz method consists of a set of consecutive projections in their natural order, that is,

$$\begin{cases} \mathbf{p}_0 &= \mathbf{x}_n \\ \mathbf{p}_i &= \mathbf{p}_{i-1} + \dfrac{b_i - \langle \mathbf{p}_{i-1}, \mathbf{a}_i \rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i\,, \quad i = 1, ..., M \\ \mathbf{x}_{n+1} &= \mathbf{p}_M \end{cases}$$

  The convergence of Kaczmarz method is discussed in [17] where the authors also give interesting projection properties.

- **Symmetric Kaczmarz method** [9]. This variant uses the classical Kaczmarz method (5.3) once for $i = 1, \ldots, M$ and then applies the same method using the equations in reverse order, i.e. for $i = M - 1, \ldots, 3, 2$. Thus, one iteration of the symmetric Kaczmarz method consists of $2M - 2$ steps.

- **Randomized Kaczmarz method** [65]. This methods differs from the classical Kaczmarz method in the way that the equations are used. We cannot talk about iterations, since every time we select the rows $\mathbf{a}_i$ randomly with probability proportional to $\|\mathbf{a}_i\|^2$. So, in order to make possible the comparison with other methods, we follow the definition proposed in [44] where one "iteration" of randomized Kaczmarz method consists of $M$ random steps.

## 5.3 Simultaneous Iterative Reconstruction Techniques (SIRT)

In the literature we find several versions of SIRT methods. Here we follow the definition given in [44, 58], where SIRT methods are written in the general form:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \lambda_n T A^T D(\underbrace{\mathbf{b} - A\mathbf{x}_n}_{\mathbf{r}_n}), \quad n = 0, 1, \ldots . \tag{5.3}$$

The vectors $\mathbf{x}_n$, $\mathbf{x}_{n+1}$ denote the current and the new iteration vector, the vector $\mathbf{r}_n = \mathbf{b} - A\mathbf{x}_n$ is the residual at the current iterate, $\lambda_n$ is a relaxation parameter, and $D \in \mathbb{R}^{M\times M}$, $T \in \mathbb{R}^{N\times N}$ are symmetric positive definite matrices. Typically, $\mathbf{x}_0$ is the zero vector or an initial guess for the solution. Concerning the convergence of SIRT memthods we have the following theorem.

**Theorem 5.1.** *[24, Theorem 6.1] The iterates of the form* (5.3) *converge to a weighted least squares solution* $\hat{\mathbf{x}}$ *of* $\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_D$ *if and only if* $0 < \lambda_n < 2/\rho(TA^TDA)$. *If in addition* $\mathbf{x}_0 \in \mathcal{R}(TA^T)$ *then* $\hat{\mathbf{x}}$ *is the unique solution of minimal Euclidean norm among all weighted least squares solutions.*

Different SIRT methods depend on the choice of the matrices $T$ and $D$.

- **Component Averaging (CAV)**. Let $s_j$ be the number of nonzero elements in the column $j$ of the matrix $A$. We define the diagonal matrix $S = \mathrm{diag}(s_1, \ldots, s_N)$

and the norm $\|\mathbf{a}_i\|_S^2 = \langle \mathbf{a}_i, S\mathbf{a}_i\rangle = \sum_{i=1}^M a_{ij}^2 s_j$, for $i = 1, \ldots, M$. By setting $T = I$ and $D = D_S = \text{diag}\left(\frac{m_i}{\|\mathbf{a}_i\|_S^2}\right)_{i=1}^M$, we obtain the method called CAV [26].

- **Diagonally Relaxed Orthogonal Projections (DROP)**. If we consider (5.3) with $T = \text{diag}\,(1/\mathrm{s_j})$ and $D = \text{diag}\left\{\frac{\mathrm{m_1}}{\|\mathbf{a}_1\|^2}, \frac{\mathrm{m_2}}{\|\mathbf{a}_2\|^2}, \ldots, \frac{\mathrm{m_M}}{\|\mathbf{a}_\mathrm{M}\|^2}\right\}$, we obtain the fully simultaneous DROP (Diagonally Relaxed Orthogonal Projections) method for linear equations. For an extensive study of this method including an analysis of the convergence the interested reader may consult [25].

- **Simultaneous Algebraic Reconstruction Technique (SART)**. If we denote by $D_r$, $D_c$ the diagonal matrices defined in terms of the row and the column sums respectively, and we set $T = D_r^{-1}$, $D = D_c^{-1}$, we have created SART. The convergence of SART is guaranteed for $0 < \lambda_n < 2$ (see [24, 45]).

In the following we always assume $T = I$ and $\lambda_n = \lambda$ fixed (stationary SIRT methods), that is

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \lambda A^T D(\mathbf{b} - A\mathbf{x}_n)\,, \quad n = 0, 1, \ldots. \tag{5.4}$$

Defining $\bar{A} = D^{1/2}A$ and $\bar{\mathbf{r}}_n = D^{1/2}(\mathbf{b} - A\mathbf{x}_n)$, we can write (5.4) as follows

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \lambda \bar{A}^T \bar{\mathbf{r}}_n\,. \tag{5.5}$$

By using the thin SVD of $\bar{A}$, given by $\bar{A} = \bar{U}\bar{\Sigma}\bar{V}^T$, with $\bar{\Sigma} \in \mathbb{R}^{N\times N}$, one can also show by induction that

$$\mathbf{x}_n - \mathbf{x}_0 = \bar{V}\bar{\Phi}^{(n)}\bar{\Sigma}^{-1}\bar{U}^T\bar{\mathbf{r}}_0\,,$$

where

$$\bar{\Phi}^{(n)} = \sum_{j=0}^{n-1}\left(I - \lambda\bar{\Sigma}^2\right)^j\left(\lambda\bar{\Sigma}^2\right) = I - \left(I - \lambda\bar{\Sigma}^2\right)^n\,. \tag{5.6}$$

Therefore, SIRT methods (5.5) can be regarded as filtering methods. It can be easily proved that SIRT methods converge to a solution of (5.1) if and only if

$$0 < \lambda < 2/\|\bar{A}\|^2 \tag{5.7}$$

The next two SIRT methods are special cases of (5.4).

- **Landweber method**. If we set $D = I$ we derive the classical Landweber method [47].

- **Cimmino method**. The original method of Cimmino [27] is recovered for $D = \frac{2}{\mu}\text{diag}\left(\frac{m_i}{\|\mathbf{a}_i\|_2^2}\right)_{i=1}^{M}$, with $\mu = \sum_{i=1}^{M} m_i$, where the $m_i$'s are arbitrary positive quantities. Therefore, Cimmino method can be regarded as a left-preconditioned version of Landweber method. Note that here $\lambda = 2$ is hidden in $D$.

## 5.4 The original method of Cimmino

The main advantage of this method is described in the introductory note of the original paper by Cimmino [27] where Mauro Picone writes that this method "is most worthy of consideration in the applications because of its generality, its efficiency and, finally, because of its guaranteed convergence which can make the method practicable in many cases" [1]. In this section we explain Cimmino's method in details and we present it in a different notation. Furthermore, we consider a $M \times N$ linear system while Cimmino in his original paper had treated the square case. An interesting and detailed report on the work of Gianfranco Cimmino is presented in [4].

Once more we want to approximate the solution $\hat{\mathbf{x}}$ of the linear system (5.1). Initially, we assume that the matrix of the system is nonsingular. We consider the set of hyperplanes (5.2) and in order to determine the common point $O \equiv \hat{\mathbf{x}}$, we define a series of approximations. Taken, as the first approximation, an arbitrary point $P_0 \equiv \mathbf{x}_0 \in \mathbb{R}^N$, we consider its symmetric points with respect to the hyperplanes (5.2), for $i = 1, ..., M$, that is

$$p_{0,i} = \mathbf{x}_0 + 2\frac{b_i - \langle \mathbf{a}_i, \mathbf{x}_0 \rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i. \tag{5.8}$$

Now, we fix $M$ arbitrary positive quantities $m_1, m_2, ..., m_M$ and we define the centroid of the system formed by placing the masses (or weights) $m_i$ at the $M$ points (5.8). This

[1] The Italian original reads: "metodo che, secondo il mio avviso, è degnissimo di essere tenuto presente nelle applicazioni e per la sua grande generalità e per la rapidità di calcolo numerico delle successive approssimazioni, ed, infine, per la sua assicurata convergenza che, in molti casi, può dare al metodo il necessario carattere di praticità"

brings us to the second approximation $P_1 \equiv \mathbf{x}_1$, given by

$$\mathbf{x}_1 = \frac{1}{\mu}\sum_{i=1}^{M} m_i p_{0,i} = \mathbf{x}_0 + \frac{2}{\mu}\sum_{i=1}^{M} m_i \frac{b_i - \langle \mathbf{a}_i, \mathbf{x}_0\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i \tag{5.9}$$

where $\mu = \sum_{i=1}^{M} m_i$. We repeat the procedure starting from the new approximation $\mathbf{x}_1$, and so forth. In this way, we obtain the following successive approximations

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{2}{\mu}\sum_{i=1}^{M} m_i \frac{b_i - \langle \mathbf{a}_i, \mathbf{x}_n\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i. \tag{5.10}$$

**Remark 5.1.** *The initial point $P_0 \equiv \mathbf{x}_0$ and its reflections* (5.8) *lie on the hypersphere with center the point $O$, namely, the solution $\hat{\mathbf{x}}$ of the linear system. Since the center of gravity $P_1$ given by* (5.9) *will necessarily fall inside the same hypersphere, it follows that it will be closer to the point $O$, that we are looking for, than the starting point $P_0$. Therefore, the point $P_{n+1} \equiv \mathbf{x}_{n+1}$ given by* (5.10) *will always fall closer to $O$. In mathematical terms, this means*

$$\|\mathbf{x}_{n+1} - \hat{\mathbf{x}}\| < \|\mathbf{x}_n - \hat{\mathbf{x}}\|.$$

The last remark shows that, in the nonsingular case, the method of Cimmino always converges to the unique solution of the system. However, we have convergence also for singular systems provided that they are consistent and the characteristic of the matrix is greater than 1. Indeed, Cimmino stated and proved the following theorem.

**Theorem 5.2.** *If the system* (5.1) *is consistent and* $\operatorname{rank}(A) \geq 2$*, then the sequence $\{\mathbf{x}_n\}$ generated by* (5.10) *converges to a solution of the system.*

**Proof.** Being $\hat{\mathbf{x}}$ the solution of the system (5.1) or equivalently of (5.2), from (5.10) we have

$$\mathbf{x}_{n+1} - \hat{\mathbf{x}} = \mathbf{x}_n - \hat{\mathbf{x}} - \frac{2}{\mu}\sum_{i=1}^{M} m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i. \tag{5.11}$$

Then,

$$\overline{OP}^2_{n+1} = \|\mathbf{x}_{n+1} - \hat{\mathbf{x}}\|^2$$

$$= \|\mathbf{x}_n - \hat{\mathbf{x}}\|^2 - \frac{4}{\mu}\langle \mathbf{x}_n - \hat{\mathbf{x}}, \sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i\rangle + \frac{4}{\mu^2}\langle \sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - hat\mathbf{x}\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i, \sum_{j=1}^M m_j \frac{\langle \mathbf{a}_j, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_j\|^2}\mathbf{a}_j\rangle$$

$$= \|\mathbf{x}_n - \hat{\mathbf{x}}\|^2 - \frac{2}{\mu^2}2\mu\langle \mathbf{x}_n - \hat{\mathbf{x}}, \sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i\rangle + \frac{2}{\mu^2}2\langle \sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i, \sum_{j=1}^M m_j \frac{\langle \mathbf{a}_j, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_j\|^2}\mathbf{a}_j\rangle$$

$$= \|\mathbf{x}_n - \hat{\mathbf{x}}\|^2 - \frac{2}{\mu^2}(2\mu\langle \mathbf{x}_n - \hat{\mathbf{x}}, \sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i\rangle - 2\langle \sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i, \sum_{j=1}^M m_j \frac{\langle \mathbf{a}_j, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_j\|^2}\mathbf{a}_j\rangle)$$

$$= \|\mathbf{x}_n - \hat{\mathbf{x}}\|^2 - \frac{2}{\mu^2}(2\mu\sum_{i=1}^M\langle \mathbf{x}_n - \hat{\mathbf{x}}, m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i\rangle - 2\sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\langle \mathbf{a}_i, \sum_{j=1}^M m_j \frac{\langle \mathbf{a}_j, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_j\|^2}\mathbf{a}_j\rangle)$$

$$= \|\mathbf{x}_n - \hat{\mathbf{x}}\|^2 - \frac{2}{\mu^2}(2\mu\sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2}\langle \mathbf{x}_n - \hat{\mathbf{x}}, \mathbf{a}_i\rangle - 2\sum_{i=1}^M m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2} \sum_{j=1}^M m_j \frac{\langle \mathbf{a}_j, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_j\|^2}\langle \mathbf{a}_i, \mathbf{a}_j\rangle)$$

$$= \|\mathbf{x}_n - \hat{\mathbf{x}}\|^2 - \frac{2}{\mu^2}(2\mu\sum_{i=1}^M m_i \frac{(\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle)^2}{\|\mathbf{a}_i\|^2} - 2\sum_{i=1}^M \sum_{j=1}^M \langle \mathbf{a}_i, \mathbf{a}_j\rangle m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2} m_j \frac{\langle \mathbf{a}_j, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_j\|^2})$$

$$= \|\mathbf{x}_n - \hat{\mathbf{x}}\|^2$$
$$-\frac{2}{\mu^2}(\mu\sum_{i=1}^M m_i \frac{(\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle)^2}{\|\mathbf{a}_i\|^2} + \mu\sum_{j=1}^M m_j \frac{(\langle \mathbf{a}_j, \mathbf{x}_n - \hat{\mathbf{x}}\rangle)^2}{\|\mathbf{a}_j\|^2} - 2\sum_{i=1}^M \sum_{j=1}^M \langle \mathbf{a}_i, \mathbf{a}_j\rangle m_i \frac{\langle \mathbf{a}_i, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_i\|^2} m_j \frac{\langle \mathbf{a}_j, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_j\|^2})$$

$$= \|\mathbf{x}_n - \hat{\mathbf{x}}\|^2 - \frac{2}{\mu^2}(\sum_{j=1}^M m_j \sum_{i=1}^M m_i\chi_i^2 + \sum_{i=1}^M m_i \sum_{j=1}^M m_j\chi_j^2 - 2\sum_{i=1}^M \sum_{j=1}^M \vartheta_{ij} m_i\chi_i m_j\chi_j).$$

Therefore,

$$\overline{OP}^2_{n+1} = \overline{OP}^2_n - \frac{2}{\mu^2}\sum_{i=1}^M\sum_{j=1}^M m_i m_j \left(\chi_i^2 + \chi_j^2 - 2\vartheta_{ij}\chi_i\chi_j\right), \tag{5.12}$$

where

$$\chi_k = \frac{\langle \mathbf{a}_k, \mathbf{x}_n - \hat{\mathbf{x}}\rangle}{\|\mathbf{a}_k\|} \ , \quad \vartheta_{ij} = \frac{\langle \mathbf{a}_i, \mathbf{a}_j\rangle}{\|\mathbf{a}_i\|\|\mathbf{a}_j\|} \, .$$

If $\chi_i\chi_j \geq 0$ then $\chi_i\chi_j =\mid \chi_i\chi_j \mid \geq \mid \vartheta_{ij} \mid\mid \chi_i\chi_j \mid = \mid \vartheta_{ij}\chi_i\chi_j \mid \geq \vartheta_{ij}\chi_i\chi_j$, as from Cauchy-Schwarz inequality $\mid \vartheta_{ij} \mid \leq 1$. Since $(\chi_i - \chi_j)^2 \geq 0$ implies $\chi_i^2 + \chi_j^2 \geq 2\chi_i\chi_j$, we finally have $\chi_i^2 + \chi_j^2 \geq 2\vartheta_{ij}\chi_i\chi_j$, thus $\chi_i^2 + \chi_j^2 - 2\vartheta_{ij}\chi_i\chi_j \geq 0$. If $\chi_i\chi_j < 0$, then $\chi_i\chi_j = - \mid \chi_i\chi_j \mid \leq - \mid \vartheta_{ij} \mid\mid \chi_i\chi_j \mid = - \mid \vartheta_{ij}\chi_i\chi_j \mid \leq -\vartheta_{ij}\chi_i\chi_j$, and finally we have $\chi_i^2 + \chi_j^2 - 2\vartheta_{ij}\chi_i\chi_j \geq \chi_i^2 + \chi_j^2 + 2\chi_i\chi_j = (\chi_i + \chi_j)^2 \geq 0$. By hypothesis, $m_i, m_j$, $i, j = 1, \ldots, M$, are positive, thus we conclude that

$$\overline{OP}^2_{n+1} \leq \overline{OP}^2_n. \tag{5.13}$$

□

**Remark 5.2.** *The necessity of the condition* rank$(A) \geq 2$ *is evident since, if all the hyperplanes* (5.2) *coincide in a single hyperplane, then the successive approximations* (5.10) *will provide alternately the starting point and its symmetric points with respect to the hyperplanes.*

**Proposition 5.1.** *The equality in* (5.13) *holds if* $\mathbf{x}_{n+1} = \mathbf{x}_n$.

**Proof.** Given that $\chi_i^2 + \chi_j^2 - 2\vartheta_{ij}\chi_i\chi_j \geq (\mid \chi_i \mid - \mid \chi_j \mid)^2$, the only way to have $\chi_i^2 + \chi_j^2 - 2\vartheta_{ij}\chi_i\chi_j = 0$ is $\chi_k$ being independent of $k$, let $\chi_k = c$ for all $k$. Then from (5.12) results

$$\overline{OP}^2_{n+1} = \overline{OP}^2_n - \frac{2}{\mu^2}\sum_{i=1}^{N}\sum_{j=1}^{N} 2c^2(1-\vartheta_{ij})m_i m_j \leq \overline{OP}^2_n - \frac{4c^2}{\mu^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(1- \mid \vartheta_{ij} \mid) m_i m_j\,.$$

If $\mid \vartheta_{ij} \mid = 1$ then (5.13) would hold as an equality. But $\mid \vartheta_{ij} \mid$ is equal to 1 only if $\mathbf{a}_i, \mathbf{a}_j$ are linearly dependent. This cannot be true for every pair $i$ and $j$ $(i, j = 1, ..., M)$ since we have assumed that rank$(A) \geq 2$. So, the only case where (5.13) is valid with "=" is if $\chi_k = 0$ for every $k = 1, ..., M$. This implies that $\langle \mathbf{a}_k, \mathbf{x}_n - \hat{\mathbf{x}} \rangle = 0$, thus $\langle \mathbf{a}_k, \mathbf{x}_n \rangle = \langle \mathbf{a}_k, \hat{\mathbf{x}} \rangle = b_k$, for all $k = 1, ..., M$, that is $\mathbf{x}_n$ verifies the system (5.2). As a consequence, the sum in (5.10) is zero and therefore $\mathbf{x}_{n+1} = \mathbf{x}_n$.

□

**Corollary 5.1.** *[27] It holds* $\overline{OP}^2_{n+1} < \overline{OP}^2_n$, *for every* $n$, *unless after a finite number of approximations we find the exact solution* $\hat{\mathbf{x}}$.

**Proposition 5.2.** *The limit of the sequence of the points* $P_n$ *is unique.*

**Proof.** All the points $P_n$ are inside the hypersphere of center $O$ and radius $\overline{OP}_0$, consequently, the sequence $\{P_n\}$ has an accumulation point, let $P \equiv \tilde{\mathbf{x}}$. We suppose that a subsequence $\{P_{n_s-1}\}$, $s = 1, 2, ...$, extracted from $\{P_n\}$ converges to $P$, thus (5.11) implies that the sequence $\{P_{n_s}\}$, $s = 1, 2, ...$, will also converge, more precisely towards the point $P_* \equiv \tilde{\mathbf{x}}_*$ defined by

$$\tilde{\mathbf{x}}_* - \hat{\mathbf{x}} = \tilde{\mathbf{x}} - \hat{\mathbf{x}} - \frac{2}{\mu}\sum_{i=1}^{M} m_i \frac{\langle \mathbf{a}_i, \tilde{\mathbf{x}} - \hat{\mathbf{x}} \rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i. \tag{5.14}$$

Repeating for (5.14) the arguments that we used for (5.10), we conclude that $\tilde{\mathbf{x}}$ is a solution of (5.2). Indeed, (5.14) implies that $\overline{OP}_* \leq \overline{OP}$. So, if we assume that $\tilde{\mathbf{x}}$ is not a solution of (5.2) then it should hold the inequality $\overline{OP}_* < \overline{OP}$, thus for $r$ and $s$ big enough, $\overline{OP}_{n_r} < \overline{OP}_{n_s-1}$, since $\{P_{n_r}\} \to P_*$ and $\{P_{n_s-1}\} \to P$. But this is impossible, as when $r = s-1$, then $n_r = n_{s-1} \leq n_s - 1$, thus, according to (5.13), it should be $\overline{OP}_{n_r} \geq \overline{OP}_{n_s-1}$. Therefore, $P = \tilde{\mathbf{x}}$ is necessarily a solution of (5.2). Now, considering that the distance $\overline{PP}_n$ will decrease as $n$ increases and that there is a sequence extracted from $\{P_n\}$ with limit the point $P$, we conclude that $P$ is the unique limit of the sequence $\{P_n\}$.

□

We should mention that a theorem analogous to (5.2) holds for inconsistent systems.

**Theorem 5.3.** *[27] The successive approximations* (5.10) *converge even if the system* (5.1) *is consistent, always provided that* $\operatorname{rank}(A) \geq 2$.

## 5.5 A generalization of Cimmino's method

In the literature we find several versions of Cimmino method, some of which we revise in this section. In order to have a general formula that unifies all the existing variants, we introduce the *Generalized Cimmino method* defined as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{\lambda_n}{\mu} \sum_{i=1}^{M} m_i \frac{b_i - \langle \mathbf{a}_i, \mathbf{x}_n \rangle}{\|\mathbf{a}_i\|^2} \mathbf{a}_i, \tag{5.15}$$

where $\lambda_n$ is a sequence of positive relaxation parameters and $\mu = \sum_{i=1}^{M} m_i$ with $m_i$ positive quantities.

**Remark 5.3.** *Formula* (5.15) *reminds the version of Cimmino method included in the AIR Tools package with the difference that in [44, Equation (5)]* $\mu$ *is equal to* $M$ *and it is independent of* $m_i$.

We could describe the cycle of one iteration of the *Generalized Cimmino method* as follows

$$\begin{cases} \mathbf{p}_0 &= \mathbf{x}_n \\ \mathbf{p}_i &= \mathbf{p}_0 + \lambda_n \dfrac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2} \mathbf{a}_i\,, \quad i = 1, ..., M \\ \mathbf{x}_{n+1} &= \dfrac{1}{\mu} \sum_{i=1}^{M} m_i \mathbf{p}_i\,. \end{cases} \tag{5.16}$$

An algorithmic scheme for the *Generalized Cimmino method* is described below.

**Generalized Cimmino Algorithm**

```
Input $A \in \mathbb{R}^{M\times N}$, $\mathbf{b} \in \mathbb{R}^M$, $\mathbf{x}_0 \in \mathbb{R}^N$,
$\lambda_n$ sequence of positive parameters,
$m_1, ..., m_M \in \mathbb{R}$ positive quantities
Set  $\mu = \sum_{i=1}^{M} m_i$
for $n = 0, 1, ...$ until convergence do
    $\mathbf{p}_0 \leftarrow \mathbf{x}_n$
    Compute $\mathbf{p}_i = \mathbf{p}_0 + \lambda_n \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2} \mathbf{a}_i, \quad i = 1, ..., M$
    $\mathbf{x}_{n+1} \leftarrow \frac{1}{\mu} \sum_{i=1}^{M} m_i \mathbf{p}_i$ .
end for $n$
```

Note that if $\lambda_n = \lambda = 1$, that is we consider projections instead of reflections, we derive the so called Cimmino's projection method. Then, being

$$\mathbf{p}_i = \mathbf{p}_0 + \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2} \mathbf{a}_i,$$

the following orthogonality properties hold

*(i)* $\langle \mathbf{b} - A\mathbf{p}_i, \mathbf{e}_i \rangle = 0,$

*(ii)* $\langle \mathbf{p}_i - \mathbf{p}_0, \mathbf{x} - \mathbf{p}_i \rangle = 0.$

**Proof.**

$$(i) \quad \langle \mathbf{b} - A\mathbf{p}_i, \mathbf{e}_i \rangle = \langle \mathbf{b}, \mathbf{e}_i \rangle - \langle A\mathbf{p}_i, \mathbf{e}_i \rangle = b_i - \langle \mathbf{p}_i, A^T\mathbf{e}_i \rangle = b_i - \langle A^T\mathbf{e}_i, \mathbf{p}_i \rangle$$

$$= b_i - \langle \mathbf{a}_i, \mathbf{p}_i \rangle = b_i - \langle \mathbf{a}_i, \mathbf{p}_0 + \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i \rangle$$

$$= b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle - \langle \mathbf{a}_i, \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i \rangle$$

$$= b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle - \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2}\langle \mathbf{a}_i, \mathbf{a}_i \rangle = 0.$$

$$(ii) \quad \langle \mathbf{p}_i - \mathbf{p}_0, \mathbf{x} - \mathbf{p}_i \rangle = \langle \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i, \mathbf{x} - \mathbf{p}_i \rangle = \langle \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2}A^T\mathbf{e}_i, \mathbf{x} - \mathbf{p}_i \rangle$$

$$= \langle \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2}\mathbf{e}_i, A\mathbf{x} - A\mathbf{p}_i \rangle = \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2}\langle \mathbf{e}_i, \mathbf{b} - A\mathbf{p}_i \rangle$$

$$= \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2}\langle \mathbf{b} - A\mathbf{p}_i, \mathbf{e}_i \rangle = 0,$$

as follows from $(i)$.

□

Following the analysis done for Kaczmarz method in [17] we study in an analogous way *Generalized Cimmino* method. First we introduce some notation. For $i = 1, ..., M$

$$\begin{aligned} \boldsymbol{\alpha}_i &= \frac{A^T\mathbf{e}_i}{\|A^T\mathbf{e}_i\|^2} = \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\,, \\ \boldsymbol{\varrho}_i &= \mathbf{b} - A\mathbf{p}_i, \\ P_i &= I - \lambda_n A\boldsymbol{\alpha}_i\mathbf{e}_i^T, \\ Q_i &= A^{-1}P_iA, \quad \text{thus} \;\; Q_i = I - \lambda_n\boldsymbol{\alpha}_i\mathbf{a}_i^T. \end{aligned} \tag{5.17}$$

**Remark 5.4.** *For $\lambda_n = \lambda = 1$ the matrix $P_i$ represents the oblique projection on $\mathbf{e}_i^\perp$ along $AA^T\mathbf{e}_i$, while $Q_i$ is the rank $N-1$ orthogonal projection on $A^T\mathbf{e}_i^\perp$ along $A^T\mathbf{e}_i$. Thus, for any vector $\mathbf{y}$, $\langle P_i\mathbf{y}, \mathbf{e}_i \rangle = 0$ and $\langle Q_i\mathbf{y}, A^T\mathbf{e}_i \rangle = 0$.*

**Remark 5.5.** *For $\lambda_n = \lambda = 2$ it results $P_i^2 = I$ and $Q_i^2 = I$, which is expected as in this case we have reflections.*

**Lemma 5.1.** *Considering* (5.17), *then the following relations hold*

(*i*) $\mathbf{p}_i = Q_i\mathbf{p}_0 + \lambda_n\langle\mathbf{b},\mathbf{e}_i\rangle\boldsymbol{\alpha}_i = Q_i\mathbf{p}_0 + (I - Q_i)\mathbf{x}$

(*ii*) $\mathbf{x} - \mathbf{p}_i = \mathbf{x} - \mathbf{p}_0 - \lambda_n\boldsymbol{\alpha}_i\mathbf{e}_i^T\boldsymbol{\varrho}_0 = Q_i(\mathbf{x} - \mathbf{p}_0)$

(*iii*) $\boldsymbol{\varrho}_i = \boldsymbol{\varrho}_0 - \lambda_n\langle\boldsymbol{\varrho}_0,\mathbf{e}_i\rangle A\boldsymbol{\alpha}_i = P_i\boldsymbol{\varrho}_0$

**Proof.**

(*i*) Using the definitions (5.17), we have

$$Q_i\mathbf{p}_0 + \lambda_n\langle\mathbf{b},\mathbf{e}_i\rangle\boldsymbol{\alpha}_i = (I - \lambda_n\boldsymbol{\alpha}_i\mathbf{a}_i^T)\mathbf{p}_0 + \lambda_n b_i\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2} = \mathbf{p}_0 - \lambda_n\frac{\mathbf{a}_i\mathbf{a}_i^T}{\|\mathbf{a}_i\|^2}\mathbf{p}_0 + \lambda_n\frac{b_i}{\|\mathbf{a}_i\|^2}\mathbf{a}_i$$

$$= \mathbf{p}_0 + \lambda_n(-\mathbf{a}_i^T\mathbf{p}_0 + b_i)\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2} = \mathbf{p}_0 + \lambda_n\frac{b_i - \mathbf{a}_i^T\mathbf{p}_0}{\|\mathbf{a}_i\|^2}\mathbf{a}_i = \mathbf{p}_i$$

and

$$Q_i\mathbf{p}_0 + (I - Q_i)\mathbf{x} = Q_i\mathbf{p}_0 + \lambda_n\boldsymbol{\alpha}_i\mathbf{a}_i^T\mathbf{x} = Q_i\mathbf{p}_0 + \lambda_n\frac{\mathbf{a}_i\mathbf{a}_i^T}{\|\mathbf{a}_i\|^2}\mathbf{x} = Q_i\mathbf{p}_0 + \lambda_n\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{a}_i^T\mathbf{x}$$

$$= Q_i\mathbf{p}_0 + \lambda_n\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}b_i = Q_i\mathbf{p}_0 + \lambda_n\boldsymbol{\alpha}_i\langle\mathbf{b},\mathbf{e}_i\rangle.$$

(*ii*) By construction,

$$\mathbf{p}_i = \mathbf{p}_0 + \lambda_n\frac{b_i - \langle\mathbf{a}_i, p_0\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i = \mathbf{p}_0 + \lambda_n\mathbf{a}_i\frac{\mathbf{e}_i^T\mathbf{b} - \mathbf{e}_i^T A\mathbf{p}_0}{\|\mathbf{a}_i\|^2} = \mathbf{p}_0 + \lambda_n\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{e}_i^T(\mathbf{b} - A\mathbf{p}_0)$$

$$= \mathbf{p}_0 + \lambda_n\boldsymbol{\alpha}_i\mathbf{e}_i^T\boldsymbol{\varrho}_0,$$

thus $\mathbf{x} - \mathbf{p}_i = \mathbf{x} - \mathbf{p}_0 - \lambda_n\boldsymbol{\alpha}_i\mathbf{e}_i^T\boldsymbol{\varrho}_0$.

Furthermore, we have

$$Q_i(\mathbf{x} - \mathbf{p}_0) = (I - \lambda_n\boldsymbol{\alpha}_i\mathbf{a}_i^T)(\mathbf{x} - \mathbf{p}_0) = \mathbf{x} - \mathbf{p}_0 - \lambda_n\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{a}_i^T\mathbf{x} + \lambda_n\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{a}_i^T\mathbf{p}_0$$

$$= \mathbf{x} - \mathbf{p}_0 - \lambda_n\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}b_i + \lambda_n\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\langle\mathbf{a}_i,\mathbf{p}_0\rangle = \mathbf{x} - \mathbf{p}_0 - \lambda_n\frac{b_i - \langle\mathbf{a}_i,\mathbf{p}_0\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i$$

$$= \mathbf{x} - \mathbf{p}_i.$$

$(iii)$ By definition of $\boldsymbol{\varrho}_0$ it holds

$$\begin{aligned}
\boldsymbol{\varrho}_0 - \lambda_n \langle \boldsymbol{\varrho}_0, \mathbf{e}_i \rangle A \boldsymbol{\alpha}_i &= \mathbf{b} - A\mathbf{p}_0 - \lambda_n \langle \mathbf{b} - A\mathbf{p}_0, \mathbf{e}_i \rangle A \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2} \\
&= \mathbf{b} - A\mathbf{p}_0 - \lambda_n \left( \langle \mathbf{b}, \mathbf{e}_i \rangle - \langle A\mathbf{p}_0, \mathbf{e}_i \rangle \right) A \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2} \\
&= \mathbf{b} - A\mathbf{p}_0 - \lambda_n A \left( b_i - \langle \mathbf{p}_0, A^T \mathbf{e}_i \rangle \right) \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2} \\
&= \mathbf{b} - A\mathbf{p}_0 - \lambda_n A \frac{b_i - \langle \mathbf{p}_0, \mathbf{a}_i \rangle}{\|\mathbf{a}_i\|^2} \mathbf{a}_i \\
&= \mathbf{b} - A \left( \mathbf{p}_0 + \lambda_n \frac{b_i - \langle \mathbf{p}_0, \mathbf{a}_i \rangle}{\|\mathbf{a}_i\|^2} \mathbf{a}_i \right) \\
&= \mathbf{b} - A\mathbf{p}_i = \boldsymbol{\varrho}_i .
\end{aligned}$$

Also,

$$\begin{aligned}
P_i \boldsymbol{\varrho}_0 &= \left( I - \lambda_n A \boldsymbol{\alpha}_i \mathbf{e}_i^T \right) (\mathbf{b} - A\mathbf{p}_0) = \mathbf{b} - A\mathbf{p}_0 - \lambda_n A \boldsymbol{\alpha}_i \mathbf{e}_i^T (\mathbf{b} - A\mathbf{p}_0) \\
&= \mathbf{b} - A\mathbf{p}_0 - \lambda_n A \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2} \left( b_i - \mathbf{e}_i^T A\mathbf{p}_0 \right) = \mathbf{b} - A\mathbf{p}_0 - \lambda_n A \frac{b_i - \mathbf{a}_i^T \mathbf{p}_0}{\|\mathbf{a}_i\|^2} \mathbf{a}_i \\
&= \mathbf{b} - A \left( \mathbf{p}_0 - \lambda_n \frac{b_i - \langle \mathbf{a}_i, \mathbf{p}_0 \rangle}{\|\mathbf{a}_i\|^2} \mathbf{a}_i \right) = \mathbf{b} - A\mathbf{p}_i = \boldsymbol{\varrho}_i .
\end{aligned}$$

□

In matrix form, the generalized Cimmino method (5.15) can be expressed as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \lambda_n A^T D \left( \mathbf{b} - A\mathbf{x}_n \right) , \qquad n = 0, 1, ...,$$

where

$$D = \frac{1}{\mu} \mathrm{diag} \left\{ \frac{\mathrm{m}_1}{\|\mathbf{a}_1\|^2}, \frac{\mathrm{m}_2}{\|\mathbf{a}_2\|^2}, \dots, \frac{\mathrm{m}_\mathrm{M}}{\|\mathbf{a}_\mathrm{M}\|^2} \right\} , \quad \mu = \textstyle\sum_{\mathrm{i}=1}^{\mathrm{M}} \mathrm{m}_\mathrm{i} \, . \tag{5.18}$$

- If we consider (5.18) with $\lambda_n = \lambda = 2$ we obtain the original method of Cimmino. If we also set $m_i = \|\mathbf{a}_i\|^2$ we obtain a special case of von Mises' method (stationary

Richardson iteration) applied to the system of normal equations $A^T A\mathbf{x} = A^T\mathbf{b}$ [57]. Moreover, if the rows of $A$ are normalized such that $m_i = \|\mathbf{a}_i\| = 1$ for $i = 1, 2, ..., M$ then $\mu = M$ and the method obtained is the under-relaxed Jacobi iteration for a special choice of the relaxation parameter [57].

- For $\lambda_n = \lambda = 1$ we derive Cimmino's projection method.
- If in (5.18) we set $\lambda_n = \lambda M$, where $\lambda$ is a constant, and $m_i = 1$, we obtain Landweber method.

Formula (5.18) can be written, equivalently, as $\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda_n A^T D\,(A\mathbf{x}_n - \mathbf{b})$ or

$$\mathbf{x}_{n+1} = Q\mathbf{x}_n + R\mathbf{b}\,, \tag{5.19}$$

where

$$Q = I - \lambda_n A^T DA \quad \text{and} \quad R = \lambda_n A^T D = (I-Q)A^{-1}\,. \tag{5.20}$$

**Lemma 5.2.** *For the sequence* $\{\mathbf{x}_n\}$ *generated by* (5.18) *the following relations hold*

$(i)$ $\mathbf{x} - \mathbf{x}_{n+1} = Q(\mathbf{x} - \mathbf{x}_n)$

$(ii)$ $\mathbf{x} - \mathbf{x}_n = Q^n(\mathbf{x} - \mathbf{x}_0)$

**Proof.**

$(i)$ We have

$$\begin{aligned} Q(\mathbf{x} - \mathbf{x}_n) &= (I - \lambda_n A^T DA)(\mathbf{x} - \mathbf{x}_n) = \mathbf{x} - \mathbf{x}_n - \lambda_n A^T DA\mathbf{x} + \lambda_n A^T DA\mathbf{x}_n \\ &= \mathbf{x} - \mathbf{x}_n + \lambda_n A^T D\,(\mathbf{b} - A\mathbf{x}_n) = \mathbf{x} - \mathbf{x}_{n+1}. \end{aligned}$$

$(ii)$ The second relation follows directly from $(i)$ using induction. Indeed, for $n = 0$, $(i)$ gives $\mathbf{x} - \mathbf{x}_1 = Q(\mathbf{x} - \mathbf{x}_0)$. We assume that for $n = k$ the relation $\mathbf{x} - \mathbf{x}_k = Q^k(\mathbf{x} - \mathbf{x}_0)$ holds. Then, from (5.19) we have

$$\begin{aligned} \mathbf{x} - \mathbf{x}_{k+1} &= \mathbf{x} - (Q\mathbf{x}_k + R\mathbf{b}) = \mathbf{x} - Q\left(\mathbf{x} + Q^k(\mathbf{x} - \mathbf{x}_0)\right) - R\mathbf{b} \\ &= \mathbf{x} - Q\mathbf{x} - Q^{k+1}(\mathbf{x} - \mathbf{x}_0) - R\mathbf{b} \\ &= (I - Q)\,\mathbf{x} - Q^{k+1}(\mathbf{x} - \mathbf{x}_0) - (I - Q)\,A^{-1}\mathbf{b} \\ &= Q^{k+1}(\mathbf{x} - \mathbf{x}_0). \end{aligned}$$

□

**Lemma 5.3.** *(i) If* $\mathbf{x} \in \mathcal{N}(A)$ *then* $Q\mathbf{x} = \mathbf{x} \in \mathcal{N}(A)$

*(ii) If* $\mathbf{x} \in \mathcal{R}(A^T)$ *then* $Q\mathbf{x} \in \mathcal{R}(A^T)$

*(iii) For any* $\mathbf{y} \in \mathbb{R}^M$, $R\mathbf{y} \in \mathcal{R}(A^T)$

**Proof.**

*(i)* By definition, $\mathbf{x} \in \mathcal{N}(A)$ implies that $A\mathbf{x} = 0$. Therefore,

$$Q\mathbf{x} = \big(I - \lambda_n A^T DA\big)\,\mathbf{x} = \mathbf{x} - \lambda_n A^T DA\mathbf{x} = \mathbf{x} \in \mathcal{N}(A).$$

*(ii)* If $\mathbf{x} \in \mathcal{R}(A^T)$ then there exist a vector $\mathbf{z} \in \mathbb{R}^M$ such that $A^T\mathbf{z} = \mathbf{x}$. So, we have

$$Q\mathbf{x} = \mathbf{x} - \lambda_n A^T DA\mathbf{x} = A^T\mathbf{z} - A^T\left(\lambda_n DA\mathbf{x}\right) = A^T\left(\mathbf{z} - \lambda_n DA\mathbf{x}\right) \in \mathcal{R}(A^T).$$

*(iii)* For any $\mathbf{y} \in \mathbb{R}^M$ we have

$$R\mathbf{y} = \lambda_n A^T D\mathbf{y} = A^T\left(\lambda_n D\mathbf{y}\right) \in \mathcal{R}(A^T).$$

□

**Proposition 5.3.** *The operators* $Q$ *and* $R$ *can be also expressed as*

$$Q = \sum_{i=1}^{M} \frac{m_i}{\mu} Q_i\,, \quad R = \sum_{i=1}^{M} \frac{m_i}{\mu} A^{-1}(I - P_i) \tag{5.21}$$

*respectively, where* $Q_i$ *and* $P_i$ *are defined in* (5.17).

**Proof.** We start with the remark that the diagonal matrix with elements $\{\dfrac{m_i}{\|\mathbf{a}_i\|^2}\}_{i=1}^M$ can be written as $\sum_{i=1}^M \dfrac{m_i}{\|\mathbf{a}_i\|^2}\mathbf{e}_i\mathbf{e}_i^T$. Then, it is immediate that

$$A^T D = \frac{1}{\mu}\sum_{i=1}^{M} m_i \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{e}_i^T\,, \quad \text{and} \quad \mathrm{A^T DA} = \frac{1}{\mu}\sum_{\mathrm{i=1}}^{\mathrm{M}} \mathrm{m_i}\frac{\mathbf{a}_\mathrm{i}}{\|\mathbf{a}_\mathrm{i}\|^2}\mathbf{a}_\mathrm{i}^\mathrm{T}.$$

From (5.17) we have $Q_i = A^{-1}P_iA$ and $P_i = I - \lambda_n A\boldsymbol{\alpha}_i\mathbf{e}_i^T$ thus

$$\begin{aligned}\sum_{i=1}^{M}\frac{m_i}{\mu}Q_i &= \sum_{i=1}^{M}\frac{m_i}{\mu}A^{-1}P_iA = \sum_{i=1}^{M}\frac{m_i}{\mu}A^{-1}\left(I - \lambda_n A\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{e}_i^T\right)A \\ &= \sum_{i=1}^{M}\frac{m_i}{\mu}A^{-1}IA - \sum_{i=1}^{M}\frac{m_i}{\mu}A^{-1}\lambda_n A\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{e}_i^T A = I - \frac{\lambda_n}{\mu}\sum_{i=1}^{M}m_i\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{a}_i^T \\ &= I - \frac{\lambda_n}{\mu}\sum_{i=1}^{M}m_i\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{a}_i^T = I - \lambda_n A^T DA = Q\end{aligned}$$

and

$$\sum_{i=1}^{M}\frac{m_i}{\mu}A^{-1}(I - P_i) = \sum_{i=1}^{M}\frac{m_i}{\mu}A^{-1}\left(\lambda_n A\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{e}_i^T\right) = \frac{\lambda_n}{\mu}\sum_{i=1}^{M}m_i\frac{\mathbf{a}_i}{\|\mathbf{a}_i\|^2}\mathbf{e}_i^T = \lambda_n A^T D = R\,.$$

□

*Note*: The operator $Q$ as expressed in (5.21) is a generalization of the operator $T$ as defined in the equation (96) from [53].

## 5.6 A new variant of Cimmino method

We dedicate this paragraph to the special case of Generalized Cimmino with $\lambda_n$ constant equal to the dimension $M$ of the problem and $m_i = 1$. This method is expressed as follows

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \sum_{i=1}^{M}\frac{b_i - \langle\mathbf{a}_i, \mathbf{x}_n\rangle}{\|\mathbf{a}_i\|^2}\mathbf{a}_i\,, \tag{5.22}$$

or in matrix form,

$$\mathbf{x}_{n+1} = \mathbf{x}_n + A^T D(\mathbf{b} - A\mathbf{x}_n),\ n = 0, 1, \dots, \tag{5.23}$$

with $D = \operatorname{diag}\left\{\frac{1}{\|\mathbf{a}_1\|^2}, \frac{1}{\|\mathbf{a}_2\|^2}, \dots, \frac{1}{\|\mathbf{a}_{\mathrm{M}}\|^2}\right\}$.

The idea of this new variant was born by the simple thought that if instead of the average of the orthogonal projections we consider their sum, we could be led faster to the solution. Indeed, when this method converges, its convergence is faster than that of the original Cimmino method or the variant with $\lambda = 1$ (cf. Figure 5.1).

The drawback is that the proposed method does not have a guaranteed convergence. This is expected if we consult the convergence Theorem 5.1. It is evident that there is a great risk that the value $\lambda = M$ is not inside the bounds that guarantee the convergence. An example where the choice $\lambda = M$ does not lead the method to convergence is illustrated in Fig. 5.2.

However, it is quite interesting the fact that the method (5.23) performs really well for orthogonal matrices. Precisely, we state the following proposition.

**Proposition 5.4.** *If the matrix $A$ of the system* (5.1) *is orthogonal, then Generalized Cimmino with $\lambda = M$ and $m_i = 1$ it converges to the exact solution in just* 1 *iteration.*

**Proof.** If $A$ is orthogonal, then $A^T DA = I$, which implies $Q = 0$ as for the method (5.23) $Q = I - A^T DA$. Finally, Lemma 5.2($i$) results $\mathbf{x} - \mathbf{x}_{n+1} = 0$ which means that even after 1 iteration ($n = 0$) we reach the exact solution $\mathbf{x}$.

□

*Note*: An analogous result holds for Kaczmarz method (see [66, Section 4]).

**Remark 5.6.** *If we would like to check a priori whether or not the value $\lambda = M$ is inside the bounds that guarantee convergence for orthogonal matrices, the answer would be positive. Since $A$ is orthogonal, it holds $\rho(A^T DA) = \dfrac{1}{M}$, thus the value $\lambda = M$ satisfies the condition of Theorem 5.1.*

This property distinguishes the method (5.23) from Cimmino with $\lambda = 1$ or $\lambda = 2$ as the last two methods fail to achieve the immediate convergence for orthogonal matrices. Having the matrix $Q$ equal to $I - \dfrac{1}{M}A^T A$ and $I - \dfrac{2}{M}A^T A$ equivalently, they perform a slow rate of convergence even in this case. An illustrative is presented in figure 5.3.

### 5.6.1 Numerical comparisons

In this paragraph we compare the new variant ($\lambda$ equal to the size of the problem) with the original method of Cimmino ($\lambda = 2$) and Cimmino's projection method ($\lambda = 1$). The experiments have been realized using test matrices from the `Matlab` gallery. The solution was always set to $\mathbf{x} = (1, ..., 1)^T$ and the right hand side $\mathbf{b}$ was computed as the product $A\mathbf{x}$. In some cases, a white noise between $10^{-2}$ and $10^{-4}$ has been added to the vector $\mathbf{b}$. The preconditioning $DA\mathbf{x} = D\mathbf{b}$, where $D = \text{diag}(1/\|\mathbf{a}_i\|)$, often met in the literature [17], is also used here.

The first test matrix, `lesp`, is a 2000 × 2000 tridiagonal matrix with real, sensitive eigenvalues and condition number almost equal to 3. This example (see figure 5.1) confirms that when the parameter $\lambda$ takes the value of the size of the problem, in this case $\lambda = 2000$, then the method performs really well. Even in the noisy case the norm of the error attains the level of the noise. On the other hand, Cimmino method with $\lambda = 1$ and $\lambda = 2$ fail to converge, having their errors stagnated at $10^1$.

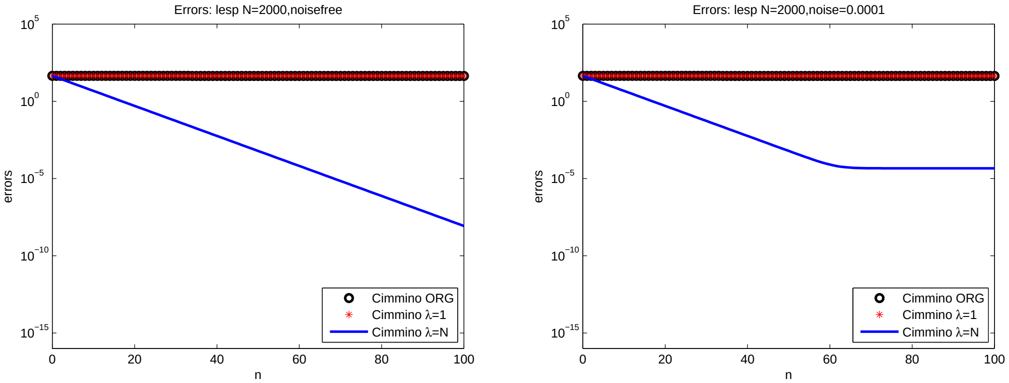


Figure 5.1: Comparison of the errors for Cimmino method with different values of the parameter $\lambda$; $\lambda = 1$, $\lambda = 2$, $\lambda = N$ for `lesp` matrix. On the left the results are noise-free, on the right we have added noise $10^{-4}$.

The results are almost reversed for `circul` matrix, a 1000 × 1000 circulant matrix with condition number at the order of $10^3$. As we see in figure 5.2 both the values $\lambda = 1$ and $\lambda = 2$ give a fast rate of convergence. What is quite surprising is that even after adding a noise equal to $10^{-2}$ we still obtain errors at the level of $10^{-7}$. On the other hand, the choice of $\lambda = N$ leads to divergence. Performing the test on the values of $\lambda$ that ensure the convergence (AIR Tools package [44]) we confirmed that the value $\lambda = 1000$ was quite out of the bounds and thus the divergence appeared.

Of a special interest are the results shown in figure 5.3. Here we have the errors of an orthogonal matrix, namely a random, orthogonal upper Hessenberg matrix (`randhess`). As we expected after the theoretical analysis, for orthogonal matrices the *Generalized Cimmino* method with $\lambda = N$ and $m_i = 1$ converges immediately both for the noise-free and the noisy case. At the same time, the other two methods still perform the usual

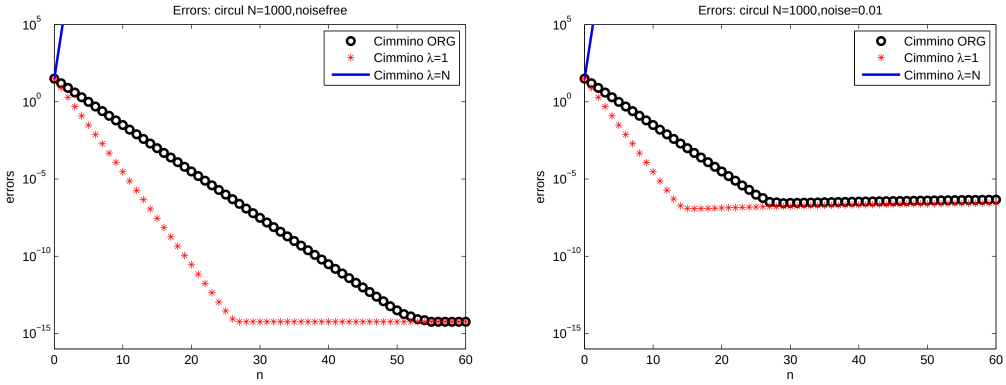


Figure 5.2: Comparison of the errors for Cimmino method with different values of the parameter $\lambda$; $\lambda = 1$, $\lambda = 2$ (original method), $\lambda = N = 1000$ for `circul` matrix. On the left the results are noise-free, on the right we have added noise $10^{-2}$.

stagnation.

## 5.7 Projected SIRT methods

**Notation**. In this paragraph we denote the $i$th component of a vector $\mathbf{z} \in \mathbb{R}^N$ by $[\mathbf{z}]_i$, $i = 1, \ldots, N$.

Various constraints on the solution $\mathbf{x}$ of (5.1) can be naturally incorporated when considering SIRT iterative solvers (5.3). When dealing with imaging problems, a typical constraint is nonnegativity [7], i.e., $[\mathbf{x}]_i \geq 0$, $i = 1, \ldots, N$. We obtain a nonnegative approximate solution by projecting $\mathbf{x}_{n+1}$ onto the nonnegative orthant of $\mathbb{R}^N$, i.e.,

$$\mathbf{x}_{n+1} = \mathcal{P}\left(\mathbf{x}_n + \lambda_n T A^T D \mathbf{r}_n\right), \quad n = 0, 1, ...,$$

where $[\mathcal{P}(\mathbf{z})]_i = [\mathbf{z}]_i$ if $[\mathbf{z}]_i \geq 0$ and $[\mathcal{P}(\mathbf{z})]_i = 0$ otherwise. The above vector can be alternatively expressed as

$$\mathbf{x}_{n+1} = P_{n+1}\widetilde{\mathbf{x}}_{n+1}, \quad \text{where} \quad \widetilde{\mathbf{x}}_{n+1} = \mathbf{x}_n + \lambda A^T \mathbf{r}_n, \tag{5.24}$$

and $P_{n+1} = \mathrm{diag}(p_1, \ldots, p_N)$, with $p_i = [\widetilde{\mathbf{x}}_{n+1}]_i$ if $[\widetilde{\mathbf{x}}_{n+1}]_i \geq 0$ and $p_i = 0$ otherwise. Another physically meaningful constraint for imaging problems is the conservation of volume $v$, defined as

$$v = \sum_{i=1}^{N} [\mathbf{x}]_i.$$

We impose both nonnegativity and conservation of volume by taking, for $i = 1, \dots, N$,

$$[\mathbf{x}_{n+1}]_i = \max([\widetilde{\mathbf{x}}_{n+1}]_i - \eta, 0)\,, \quad \text{where} \quad \sum_{i=1}^{N} \max([\widetilde{\mathbf{x}}_{n+1}]_i - \eta, 0) = v\,, \tag{5.25}$$

and $\widetilde{\mathbf{x}}_{n+1}$ is defined as in (5.24). The above approach relies on Lagrange multipliers (i.e., the scalar $\eta > 0$), and the value of $v$ is updated at each iteration by taking the sum of the entries of $\mathbf{x}_n$, the initial value of $v$ being the sum of the entries of $\mathbf{b}$.

## 5.8 Semi-iterative methods

The iterative regularization methods revised in the present chapter are characterized by a slow rate of convergence. In the past decades, several strategies have been proposed to accelerate these methods. Here we focus on the acceleration of Landweber methods succeeded by the so-called semi-iterative methods (see, for instance, [34, 40]). These methods combine one Landweber iteration with an averaging process over all or some of the previous approximations of the solution. An example of semi-iterative methods are $\nu$-methods (cf. [34, §6.2 and 6.3], [40], and the references therein), which rely on a three-term update formula. The $\nu$-methods were originally introduced by Brachage [12] to obtain theoretical estimates for the performance of the conjugate gradient method. Nowadays, $\nu$-methods are considered as an alternative to the conjugate gradient method and the classical Landweber method.

At the $n$-th iteration, $n \geq 2$, $\nu$-methods can be expressed as

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mu_n(\mathbf{x}_{n-1} - \mathbf{x}_{n-2}) + \lambda_n A^T(\mathbf{b} - A\mathbf{x}_{n-1}) \tag{5.26}$$

with $\mu_1 = 0$, $\lambda_1 = (4\nu + 2)/(4\nu + 1)$ and

$$\mu_n = \frac{(n-1)(2n-3)(2n+2\nu-1)}{(n+2\nu-1)(2n+4\nu-1)(2n+2\nu-3)}\,, \tag{5.27}$$

$$\lambda_n = 4\frac{(2n+2\nu-1)(n+\nu-1)}{(n+2\nu-1)(2n+4\nu-1)}, \qquad n > 1 \tag{5.28}$$

Depending on the regularity of the exact solution $\hat{\mathbf{x}}$, and the choice of $\nu$, it can be showed that a $\nu$-method is an order-optimal regularization method.

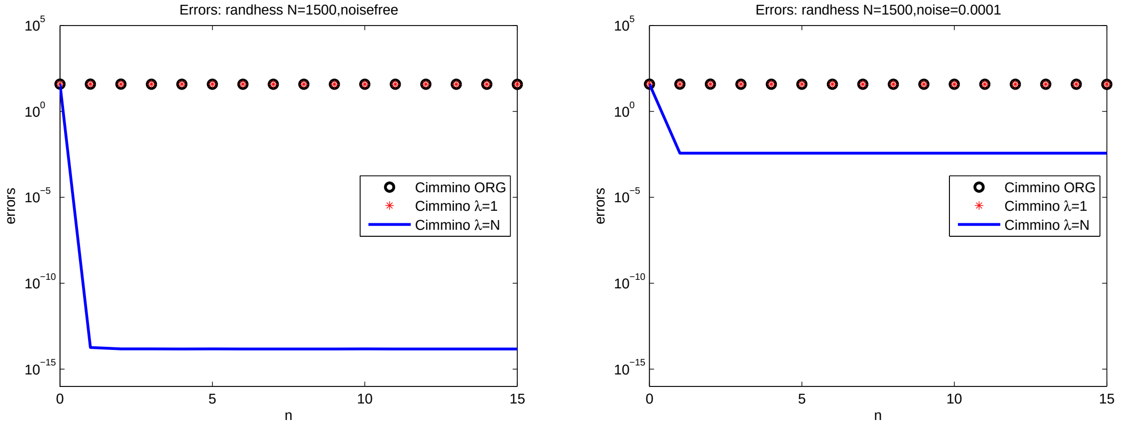


Figure 5.3: Comparison of the errors for Cimmino method with different values of the parameter $\lambda$; $\lambda = 1$, $\lambda = 2$ (original method), $\lambda = N = 1500$ for `randhess` matrix. On the left the results are noise-free, on the right we have added noise.

# Chapter 6

# Convergence acceleration using the simplified topological $\varepsilon$-algorithm

In the previous chapter we revised the iterative regularization methods which are commonly used for the solution of large-scale inverse ill-posed problems. In this chapter we study the convergence acceleration of these methods. More precisely, we provide an insight and algorithmic details about the simplified topological $\varepsilon$-algorithm applied to Cimmino and Landweber methods. The numerical experiments, which consist of applications in medical imaging (computerized tomography), seismic tomography and image deblurring, illustrate the gain from the use of extrapolation on different methods. We also compare the extrapolated methods with other known acceleration methods and some Krylov subspace methods.

## 6.1 Presentation

Fundamentally, tomographic imaging deals with reconstructing an image from its projections. A continuous model for this problem is described by an integral equation of the first kind along lines (rays), and depends on the technology and geometry of the scanning device [44, 52]. In image deblurring, the goal is to restore a distorted image given the so-called point spread function (PSF), which describes the distortion of each point (pixel) of the image. In the following we consider spatially invariant PSFs that are defined both experimentally and analytically. A continuous model is described by a 2D integral equation of the first kind that models the convolution process. Suitable boundary conditions that describe the behavior of the pixels at the edges of an image should be assigned [5, 43].

Upon discretization, a linear system of the form

$$A\mathbf{x} + \mathbf{e} = \mathbf{b} \tag{6.1}$$

is recovered, where $A \in \mathbb{R}^{M\times N}$ and $\mathbf{b} \in \mathbb{R}^M$ are known, and the vector $\mathbf{e} \in \mathbb{R}^M$ represents unknown errors or noise in the measured data $\mathbf{b}$. We assume $\mathbf{e}$ to be Gaussian white noise. In the following we denote by $\hat{\mathbf{b}}$ the unknown error-free right-hand side vector associated to (6.1), i.e., $A\hat{\mathbf{x}} = \hat{\mathbf{b}}$; $\hat{\mathbf{x}}$ is called exact solution associated to (6.1). The ill-posed nature of the continuous problem is inherited by the discrete one.
Let us consider the singular value decomposition (SVD) of $A$, given by

$$A = U\Sigma V^T\,,$$

where $U = [\mathbf{u}_1, \dots, \mathbf{u}_M] \in \mathbb{R}^{M\times M}$ and $V = [\mathbf{v}_1, \dots, \mathbf{v}_N] \in \mathbb{R}^{N\times N}$ are orthogonal, and $\Sigma \in \mathbb{R}^{M\times N}$ consists of $\mathrm{diag}(\sigma_1, \dots, \sigma_{\min\{M,N\}})$ with post-pended zero rows or columns (if $M > N$ or $M < N$, respectively). The singular values $\sigma_i \geq \sigma_{i+1} \geq 0$ quickly and smoothly decay to zero and, correspondingly, the singular vectors $\mathbf{v}_i$ display increasing oscillations. Once the solution of (6.1) is expressed with respect to the SVD of $A$ as

$$\mathbf{x} = \sum_{i=1}^{\min\{M,N\}} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i\,,$$

we see that, because of the presence of $\mathbf{e}$ in (6.1), $\mathbf{x}$ is dominated by the high-frequency components $\sigma_i^{-1}\mathbf{v}_i$ for large indices $i$. The most popular regularization methods can be regarded as filtering methods, i.e., methods that suitably attenuate the influence of high-frequency noise components on the approximated solution. In this framework, the regularized solution of (6.1) can be generically expressed as

$$\mathbf{x}_{\text{filt}} = V\Phi\Sigma^\dagger U^T\mathbf{b} = \sum_{i=1}^{\min\{M,N\}} \phi_i\frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i\,,$$

where † denotes the Moore-Penrose pseudo inverse, and $\Phi = \mathrm{diag}(\phi_1, \dots, \phi_{\min\{M,N\}})$. Different filtering methods differ for the definition of the filter factors $\phi_i$, $i = 1, \dots, N$, but a common requirement is to have $\phi_i \simeq 1$ for $i$ small (i.e., no filtering on the low-frequency components of $\mathbf{x}_{\text{filt}}$), and $\phi_i \simeq 0$ for $i$ large (i.e., some filtering on the high-frequency components of $\mathbf{x}_{\text{filt}}$). For this reason, the more filter factors are close to zero, the smoother the reconstruction. Truncated SVD and Tikhonov regularization belong to the class of filtering methods, which typically require the SVD of $A$ to be available. Given the 2D nature of the solution $\mathbf{x} \in \mathbb{R}^N$, the linear system (6.1) is inherently large-scale. For this reason, unless $A$ has some particular structure, the SVD of $A$ is prohibitively expensive to compute. In this case, in practice, only iterative regularization can be employed, which typically rely on matrix-vector multiplications with $A$ and/or $A^T$. Iterative regularization consists in applying an appropriate iterative method to (6.1) and in terminating the iterations sufficiently early, before the high-frequency components totally corrupt the solution. This phenomenon is referred to as semi-convergence. In the following, although there are potentially no constraints on the dimensions of the linear system (6.1), we just consider the square or overdetermined cases, i.e., $M \geq N$.

In this thesis we consider regularizing iterative solvers for (6.1). We exclusively take into account the methods that are collected in the Matlab package *AIR Tools* [44] (see also [58] for a detailed description of the methods). In particular, we focus on Cimmino and Landweber methods, which are very popular and basic iterative strategies. For image deblurring problems, such basic iterative methods are usually outperformed by the more recent class of Krylov subspace methods. Indeed, Cimmino and Landweber methods are remarkably slower than any Krylov method [5, 36]. However, on the upside, when employing Landweber and Cimmino methods, physically meaningful constraints can be naturally incorporated at each iteration; the same is not true for Krylov subspace methods. A typical constraint to be imposed in imaging problems is nonnegativity; this comes naturally, since the pixel values of an image (i.e., entries of the vector $\mathbf{x}$) are known to be nonnegative. Moreover, a precise reformulation of these basic iterative methods as spectral filtering methods is available. Therefore, even in the case of image restoration problems, Landweber, Cimmino, or related methods might deliver more accurate approximations of $\hat{\mathbf{x}}$ than (and might be eventually preferred to) Krylov subspace methods.

Cimmino and Landweber methods are widely used in tomographic reconstruction and image restoration problems. Several strategies have been proposed in order to accelerate their usually slow convergence. In Section 5.8 we revised the semi-iterative methods, also called accelerated Landweber methods. An analogous idea of automatic acceleration has been proposed in [8], where the authors focus on line search methods and propose a new way to update the search direction by involving the previous approximation of the solution. This strategy has been mainly applied to the Richardson-Lucy algorithm, i.e., for problems corrupted by Poisson noise. Moreover, for this class of problems, and when simple convex constraints (such as nonnegativity and volume conservation) have to be imposed, the authors of [10] derive an efficient scaled projected gradient method. The convergence of the scaled gradient projection method is fast thanks to special scaling strategies and step length updating rules that exploit previous information.

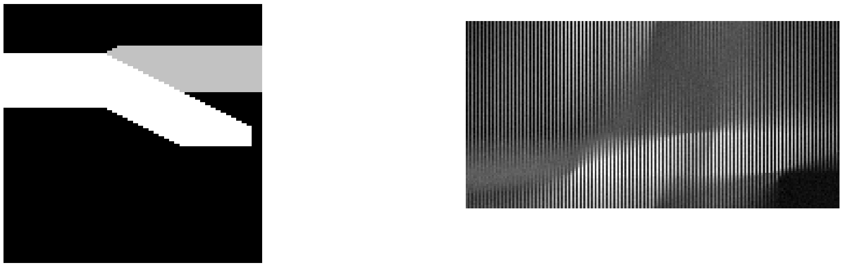
In this chapter we propose a different approach originally developed in [35]. In particular, we investigate the performance of STEA (see Section 4.3) when applied to Cimmino and Landweber methods for tomography and image deblurring problems. We mention that STEA has been already tested in [21] for accelerating the convergence of Kaczmarz method and in problems where a sequence of square matrices has to be accelerated. The authors of [6] and [22] consider a wide range of applications and iterative methods that can benefit from the use of TEA or STEA. Moreover, in [17], the authors successfully apply TEA to accelerate Kaczmarz method, mostly when applied to well-posed problems. Here we deal with ill-posed problems and we explore the convergence acceleration brought by STEA when applied to several reconstruction techniques.

We stress that throughout this chapter we always consider the third expression of STEA2, that is formula (4.16). However, some general remarks hold for any expression of the simplified topological $\varepsilon$-algorithm. Every time it will be clear from the context if the term STEA refers to the simplified topological $\varepsilon$-algorithm in general or formula (4.16) in particular.

In the sequel, we describe the image reconstruction and restoration test problems considered in the next sections. Then we study the convergence properties of certain SIRT methods accelerated by STEA and we give some insight into the extrapolated versions of these methods. Following [6, 17], we also consider the simultaneous use of extrapolation and restart strategies. Section 6.4 presents the most significant numerical experiments including comparisons between accelerated and unaccelerated version of the same iterative method as well as comparisons with other well-known methods.

## 6.2 Test problems

For the purposes of this chapter three test problems have been used. Their technical details are described below.



(a) Exact object.

(b) Set of noisy projections.

Figure 6.1: `seismictomo` test problem.

- `seismictomo`. This is a 2D seismic tomography test problem from [44]. We use the phantom shown in Figure 6.1a, which models a 2D subsurface of simple convergent boundaries of two tectonic plates with different slowness. For further information on the seismic tomography problem and a deeper understanding of the technical terms, the interested reader should consult [58, Chapter 6]. In our numerical tests, we choose a discretization equal to $n = 100$, i.e., the two-dimensional domain illustrating a cross section of the subsurface is divided into $n = 100$ equally spaced intervals in both dimensions, creating $n^2$ cells. On the right boundary $s = 100$ sources are located and each source transmits waves to the $p = 200$ seismographs or receivers, which are scattered on the surface and on the left boundary. Therefore, the overdetermined matrix $A$ has $M = ps = 20000$ rows and $N = n^2 = 10000$ columns. We create a noisy right-hand side by adding Gaussian white noise of level $\delta = {}^{\|\mathbf{e}\|}/_{\|\hat{\mathbf{b}}\|} = 5 \cdot 10^{-2}$ (Figure 6.1b).

- `cameraman`. This is an image deblurring test problem. More precisely, the exact test image has $256 \times 256$ pixels. We consider a nonsymmetric Gaussian blur, described analytically by the function

$$k(x,y) = \frac{1}{2\pi\alpha_1\alpha_2} \exp\left(-\frac{1}{2\alpha_1^2}x^2 - \frac{1}{2\alpha_2^2}y^2\right) .$$

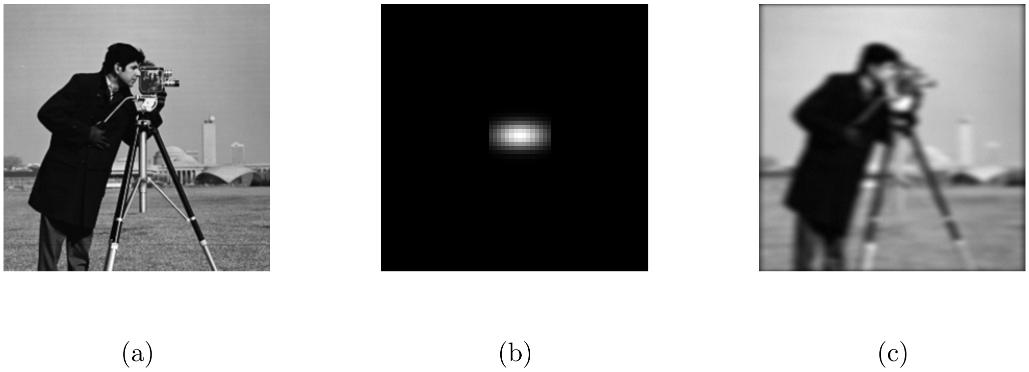


Figure 6.2: `cameraman` test problem. (a) Exact image. (b) Blow-up (400%) of the PSF. (c) Blurred and noisy image.

In our tests, we take $\alpha_1 = 4$, $\alpha_2 = 2$, and we only consider $x, y \in \{0, \dots, 7\}$. We impose zero boundary conditions, so that $A$ is a sparse square Toeplitz block matrix with Toeplitz blocks. The size of $A$ is $N = 256^2 = 65536$. We obtain the noisy data by adding Gaussian white noise of level $\delta = {}^{\|\mathbf{e}\|}/_{\|\hat{\mathbf{b}}\|} = 10^{-2}$. We refer to [42] and [43] for more details on analogous image deblurring test problems. In Figure 6.2 we display the exact image, the PSF, and the blurred and noisy image for this test problem.

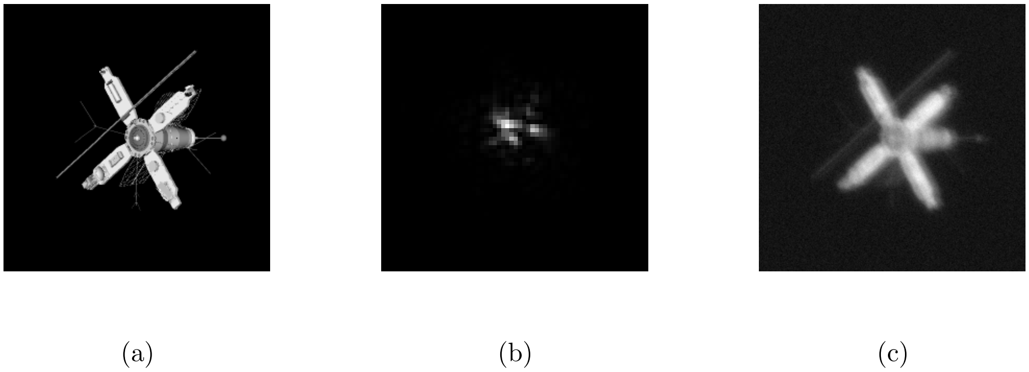


Figure 6.3: `satellite` test problem. (a) Exact image. (b) Blow-up (400%) of the PSF. (c) Blurred and noisy image, with $\delta = 10^{-1}$.

- `satellite`. This is a famous astronomical imaging test problem from *Restore*

*Tools* [51]. This test basically models the effect of atmospheric turbulence on the image of a satellite recovered by ground-based telescopes. We consider a mild blur, and we impose reflective boundary conditions. $A$ is a square matrix of size $N = 256^2 = 65536$. We refer to [5, 51] for a complete and detailed description of this test problem. The noisy data are obtained by adding Gaussian white noise of different levels, up to $\delta = \|\mathbf{e}\|/\|\hat{\mathbf{b}}\| = 10^{-1}$. In Figure 6.3 we display the exact image, the PSF, and the blurred and noisy image for this test problem.

## 6.3 STEA acceleration of iterative reconstruction techniques

In this section we present the extrapolation applied to SIRT methods but we stress that all the iterative methods presented in Chapter 5 can be treated in the same way. However, the convergence properties hold only for the method on which the analysis is based every time.

**Algorithm 1** STEA for SIRT methods

```
Input $A \in \mathbb{R}^{M\times N}$, $\mathbf{b} \in \mathbb{R}^M$, $\mathbf{x}_0 \in \mathbb{R}^N$
Compute $\mathbf{x}_1, \mathbf{x}_2$ by (5.3).
Compute $\mathbf{z}_0 = \tilde{\boldsymbol{\varepsilon}}_2^{(0)}$    (Extrapolation step)
for $n = 2, 3, \ldots$ until a stopping rule is satisfied do
    Compute $\mathbf{x}_{n+1}$ by (5.3).
    Compute $\mathbf{z}_{n-1} = \begin{cases} \tilde{\boldsymbol{\varepsilon}}_{n+1}^{(0)}, & \text{if } n \text{ is odd;} \\ \tilde{\boldsymbol{\varepsilon}}_{n}^{(1)}, & \text{if } n \text{ is even.} \end{cases}$    (Extrapolation step).
end for $n$
```

SIRT methods can be extrapolated both in a purely iterative way, or considering suitable restarts of the methods. These two approaches are summarized in Algorithm 1 and 2, respectively. In particular, the restarting technique consists in performing, in a first instance, a few iterations of the iterative method (starting from an initial vector $\mathbf{x}_0$, usually the zero vector), and applying the extrapolation method to the sequence of computed vectors (including $\mathbf{x}_0$). When a restart happens, one takes as new initial guess $\mathbf{x}_0$ the vector obtained by extrapolating the previous set of iterations. In the following,

**Algorithm 2** Restarted STEA for SIRT methods

```
**Input** $A \in \mathbb{R}^{M\times N}$, $\mathbf{b} \in \mathbb{R}^M$, $\mathbf{x}_0^{(0)} \in \mathbb{R}^N$.
**for** $j = 0, 1, \ldots$ until a stopping rule is satisfied **do**
    **for** $n = 0, ..., \ell - 1$ **do**
        **Compute** $\mathbf{x}_{n+1}^{(j)}$ by (5.3).
    **end for** $n$
    **Compute** $\mathbf{z}_j = \begin{cases} \tilde{\boldsymbol{\varepsilon}}_\ell^{(0)}, & \text{if } \ell \text{ is even;} \\ \tilde{\boldsymbol{\varepsilon}}_{\ell-1}^{(1)}, & \text{if } \ell \text{ is odd.} \end{cases}$    (Extrapolation step).
    **Take** $\mathbf{x}_0^{(j+1)} = \mathbf{z}_j$.
**end for** $j$
```

each set of iterations performed within two consecutive restarts is called cycle and, in order to keep track of the number of performed cycles, we equip $\mathbf{x}_0$ with a bracketed upper index. The total number of iterations of a restarted strategy is defined as the sum of the iterations of each cycle. We note that both the scalars $m$ (i.e., the number of cycles) and $\ell$ (i.e., the number of iterations at each cycle) can be set by considering suitable stopping criteria (cf. Section 6.4).

**Remark 6.1.** *Concerning the computational cost of Algorithm 1 and 2, we stress that STEA adds no significant overhead. Indeed, it only requires the computation of inner products (see* (4.9)*), and sums or differences of vectors, whilst the use of the scalar $\varepsilon$-algorithm adds the cost of simple operations between scalars.*

We now provide some theoretical insight on Algorithms 1 and 2. We perform just 3 (overall, or at each cycle) iterations of a SIRT method (5.5), so that we can apply STEA with $k = 0$. In this way, we obtain a quite simple expression for the extrapolated vector in (4.16). In fact, recalling expression (5.5) and applying the scalar $\varepsilon$-algorithm (1.17) to the sequence $\langle \mathbf{y}, \mathbf{x}_n \rangle$, we obtain for $k = 0$

$$\varepsilon_0^{(n)} = \langle \mathbf{y}, \mathbf{x}_n \rangle, \quad \varepsilon_1^{(n)} = \left( \langle \mathbf{y}, \lambda \bar{A}^T \bar{\mathbf{r}}_n \rangle \right)^{-1}, \text{ and } \varepsilon_2^{(n)} = \langle \mathbf{y}, \mathbf{x}_{n+1} \rangle + \frac{\langle \mathbf{y}, \bar{A}^T \bar{\mathbf{r}}_{n+1} \rangle \langle \mathbf{y}, \bar{A}^T \bar{\mathbf{r}}_n \rangle}{\langle \mathbf{y}, (\bar{A}^T \bar{A}) \bar{A}^T \bar{\mathbf{r}}_n \rangle}.$$

In the above expressions we have also exploited the relation $\bar{\mathbf{r}}_n - \bar{\mathbf{r}}_{n+1} = \lambda \bar{A} \bar{A}^T \bar{\mathbf{r}}_n$. Thanks to these derivations, the scalar ratio in formula (4.16) can be specified as

$$\frac{\varepsilon_2^{(n)} - \varepsilon_0^{(n+1)}}{\varepsilon_0^{(n+2)} - \varepsilon_0^{(n+1)}} = \frac{\langle \mathbf{y}, \bar{A}^T \bar{\mathbf{r}}_n \rangle}{\lambda \langle \mathbf{y}, \bar{A}^T \bar{A} \bar{A}^T \bar{\mathbf{r}}_n \rangle}.$$

Consequently it is easy to show that

$$\begin{aligned}\tilde{\boldsymbol{\varepsilon}}_2^{(n)} &= \mathbf{x}_{n+1} + \frac{\langle \mathbf{y}, \bar{A}^T \bar{\mathbf{r}}_n \rangle}{\lambda \langle \mathbf{y}, \bar{A}^T \bar{A} \bar{A}^T \bar{\mathbf{r}}_n \rangle} (\mathbf{x}_{n+2} - \mathbf{x}_{n+1}) && (6.2)\\ &= \mathbf{x}_{n+1} + \frac{\langle \mathbf{y}, \bar{A}^T \bar{\mathbf{r}}_n \rangle}{\langle \mathbf{y}, \bar{A}^T \bar{A} \bar{A}^T \bar{\mathbf{r}}_n \rangle} \bar{A}^T \bar{\mathbf{r}}_{n+1} && (6.3)\end{aligned}$$

Focussing on expression (6.2), we observe that STEA algorithm in this special case delivers a three-term recursion formula for computing the new approximation $\tilde{\boldsymbol{\varepsilon}}_2^{(n)}$ of the solution of (6.1); for this reason, we can say that STEA is analogous to the $\nu$-methods (5.27). Another point in common is that the performance of STEA and $\nu$-methods heavily depends on the choice of the vector $\mathbf{y} \in \mathbb{R}^N$ and the parameter $\nu > 0$, respectively. However, from a theoretical point of view, $\nu$-methods are well-understood regularization methods, and their properties are studied by exploring relations with orthogonal polynomials [40]; the same analysis has not been performed for (6.2), so far. Moreover, while $\nu$-methods are tailored to accelerate the convergence of Landweber method, STEA algorithm can be potentially applied to every sequence of arrays.

### 6.3.1 Choice of y

In the literature, there is no theoretical study on the choice of the vector $\mathbf{y}$ in (4.9). When dealing with iterative solvers for linear systems, some typical choices are $\mathbf{y} = \mathbf{e}_i$, $\mathbf{y} = (1, \ldots, 1)^T \in \mathbb{R}^N$ (in the following denoted by $\mathbf{y} = \mathbf{1}$), or a random vector of length $N$ with components normally distributed in $[-1\,, +1]$ (in the following denoted by $\mathbf{y} = \texttt{rand}$) [6, 17, 21, 22]. Despite the lack of theory on the choice of $\mathbf{y}$, a good performance of the simplified topological $\varepsilon$-algorithm depends also on $\mathbf{y}$; based on the extensive numerical tests performed with the topological $\varepsilon$-algorithm in the aforementioned papers, we conclude that a successful choice of $\mathbf{y}$ is highly dependent on both the problem and the sequence. In the following we give some insight into the choice of the vector $\mathbf{y} \in \mathbb{R}^N$ in (4.9) for Landweber method, i.e., $D = I$ in (5.5) (so that $\bar{A} = A$ and $\bar{\mathbf{r}}_n = \mathbf{r}_n$). First we consider formula (6.3) in this particular case; we remark that, at least formally, the extrapolated vector $\tilde{\boldsymbol{\varepsilon}}_2^{(n)}$ can be expressed as a vector obtained by a SIRT iterative method (5.3). Let us consider the choice $\mathbf{y} = \mathbf{r}_0^{(j)}$ in (6.3). We obtain the following expression for the extrapolated restarted Landweber vector

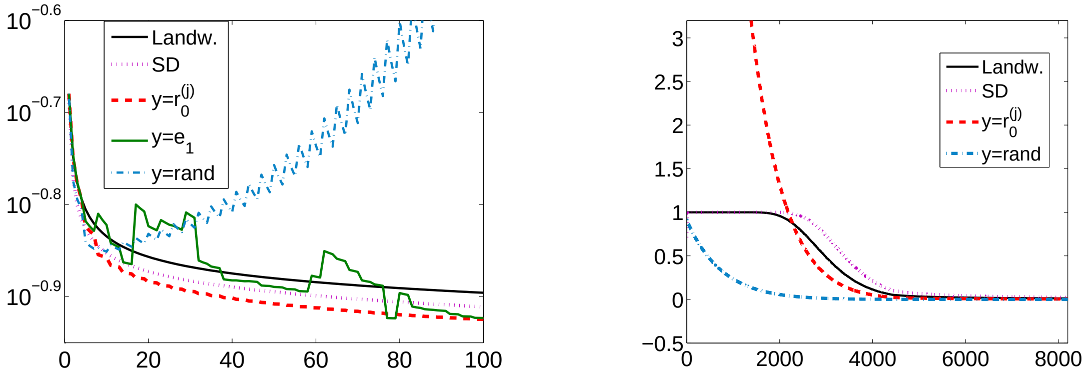
$$\tilde{\boldsymbol{\varepsilon}}_2^{(n)} = \mathbf{x}_{n+1} + \underbrace{\frac{\langle \mathbf{r}_0^{(j)}, A^T\mathbf{r}_n\rangle}{\langle \mathbf{r}_0^{(j)}, A^TAA^T\mathbf{r}_n\rangle}}_{\lambda^{extr}} A^T\mathbf{r}_{n+1}\,. \tag{6.4}$$

It is evident that the above update formula is very similar to the one obtained by applying the Steepest Descent (SD, sometimes also called residual norm SD [48]) method to (6.1). Indeed, in the SD case, one gets

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{\|A^T\mathbf{r}_n\|^2}{\|AA^T\mathbf{r}_n\|^2}A^T\mathbf{r}_n = \mathbf{x}_n + \underbrace{\frac{\langle A^T\mathbf{r}_n, A^T\mathbf{r}_n\rangle}{\langle A^T\mathbf{r}_n, A^TAA^T\mathbf{r}_n\rangle}}_{\lambda_n^{SD}} A^T\mathbf{r}_n\,; \tag{6.5}$$

in this case, $A^T\mathbf{r}_n$ is regarded as a descent direction along the negative gradient of $\|\mathbf{b}-A\mathbf{x}\|$, and the relaxation parameter $\lambda_n^{SD}$ is regarded as a step-length that minimizes $\|\mathbf{b}-A\mathbf{x}\|$ at each iteration in the direction of the negative gradient. Formula (6.4) differs from (6.5) only for different occurrences of $A$ in the step-length $\lambda^{extr}$, which is updated at each restart, and for the residual considered, i.e., in the extrapolated Landweber case the residual is the one computed at the beginning of the restarts, while in the SD case the residual is computed at the previous iteration. Since the analysis in [48] shows that SD method can be successfully applied to regularize ill-posed systems, and since SD is often more efficient than Landweber method (because of the adaptive choice of the relaxation parameter), the analogy between SD and extrapolated Landweber provides a qualitative justification of the success of the choice $\mathbf{y} = \mathbf{r}_0^{(j)}$ for the latter. We stress that the same analogy does not hold if the choices $\mathbf{y} = \mathbf{e}_i$, $\mathbf{y} = \mathbf{1}$, $\mathbf{y} = \mathbf{x}_0^{(j)}$ (with $\mathbf{x}_0^{(0)} \neq \mathbf{0}$), or $\mathbf{y} =$ `rand` are performed. Indeed, the previous choices may also result in relaxation parameters in (6.3) that are too big, and therefore compute bad approximations of $\hat{\mathbf{x}}$ at each iteration.

**An example.** In order to support the above considerations, we present a numerical experiment with the `cameraman` test problem. Note that the PSF of this test problem is separable (i.e., can be decomposed in row and column blurs, see [43]), so that we are able to compute the SVD of the blurring matrix $A$. To start with, we compare the performance of the restarted STEA for different choices of $\mathbf{y}$ when $n = 3$ steps of the Landweber method are performed at each iteration cycle; we also take into account the SD method (Fig. 6.4a). In Fig. 6.4b we plot the filter factors (5.6) associated to the (extrapolated) Landweber method and the SD method for different choices of $\mathbf{y}$. We can clearly see that the choice $\mathbf{y} = \mathbf{r}_0^{(j)}$ delivers components that are close to the usual Landweber and SD filter factors, except for the first ones (corresponding to the biggest singular values) that are quite larger. The choice $\mathbf{y} = \texttt{rand}$ delivers filter factors that are closer to zero and, therefore, it computes over-regularized solutions.



(a)

(b)

Figure 6.4: `cameraman` test problem. **(a)** Relative error history using Landweber and SD methods, and the restarted extrapolated versions of Landweber with $\mathbf{y} = \texttt{rand}$, $\mathbf{y} = \mathbf{e}_1$, and $\mathbf{y} = \mathbf{r}_0^{(j)}$. **(b)** Filter factors at the 50th (total) iteration of the Landweber and SD methods, and of the extrapolated Landweber method with $\mathbf{y} = \texttt{rand}$ and $\mathbf{y} = \mathbf{r}_0^{(j)}$. On the horizontal axis we only display the first 8000 components (after that, all the components are approximately zero); on the vertical axis, we truncate after 3 (the maximum value reached by the extrapolated Landweber method with $\mathbf{y} = \mathbf{r}_0^{(j)}$ is around 15).

Although the formal derivations are more cumbersome, we believe that the same holds when more steps of a SIRT method are performed, or when other iterative methods are applied (see the experiments in Section 6.4). Moreover, our analysis confirms that, when applying Algorithm 2, a reliable choice for the TEA or STEA algorithm is $\mathbf{y} = \mathbf{r}_0^{(j)}$. Analogous remarks have been stated in [6, 17]. More precisely, the authors of [6] consider the Gauss-Seidel method and validate the choice of $\mathbf{y} = \mathbf{r}_0^{(j)}$ by many numerical experiments; the authors of [17] consider a variant of Kaczmarz method and mention that, from a theoretical point of view, the extrapolated vectors obtained with $\mathbf{y} = \mathbf{r}_0^{(j)}$

coincide with the iterates of Lanczos-based solvers for (6.1).

### 6.3.2 Acceleration properties

Let us now study further the acceleration properties of STEA applied to a particular case of nonnegatively projected Landweber method, and for a specific choice of the vector $\mathbf{y} \in \mathbb{R}^N$. We consider formulation (5.24) and we assume $P_n = \cdots = P_{n+k} =: P$. This assumption typically holds as soon as some stabilization occurs in the approximate solutions. By exploiting an expression analogous to (5.6), with $D = I$ and $\mathbf{x}_0 = 0$ (so that $\bar{A} = A$ and $\bar{\mathbf{r}}_0 = \mathbf{b}$), it can be easily shown by induction on $k \geq 1$ that

$$(-1)^k \Delta^k \mathbf{x}_n = -P(\lambda A^T A)^{k-1}(I - \lambda A^T A)^n (\lambda A^T \mathbf{b})\,. \tag{6.6}$$

We now verify the assumptions of Theorem 4.3. By taking $\mathbf{y} = -A^T\mathbf{b}$, and exploiting (6.6) and the fact that $P(\lambda A^T A)^{k-1}(I - \lambda A^T A)^n$ is symmetric and semipositive definite (if $\lambda$ satisfies (5.7)), we get

$$\begin{array}{rcl} (-1)^k \Delta^k \left(\langle -A\mathbf{b}, \mathbf{x}_n \rangle\right) & = & \langle -A\mathbf{b}, (-1)^k \Delta^k \mathbf{x}_n \rangle \\ & = & \langle A\mathbf{b}, P(\lambda A^T A)^{k-1}(I - \lambda A^T A)^n (\lambda A^T \mathbf{b}) \rangle \geq 0\,, \end{array}$$

so that (4.18) is verified. Hypothesis (4.19) holds if we take $a = -1$, and provided that $n$ is sufficiently big. Indeed, substituting the SVD of $A$ into (6.6), we get

$$-(-1)^k \Delta^k (-\mathbf{x}_n) = PV \underbrace{(\lambda \Sigma^2)^k (I - \lambda \Sigma^2)^n}_{\Psi^{(n,k)}} \Sigma^{-1} U^T b\,,$$

so that $-(-1)^k \Delta^k (-\mathbf{x}_n)$ can be regarded as a filtered vector. Recalling that

$$\mathbf{x}_n = PV \underbrace{\left(I - (I - \lambda \Sigma^2)^n\right)}_{\Phi^{(n)}} \Sigma^{-1} U^T b \geq 0\,, \tag{6.7}$$

we can guarantee that $(-1)^k(-\Delta^k(-\mathbf{x}_n)) \geq 0$ if

$$(\lambda \sigma_i^2)^k (1 - \lambda \sigma_i^2)^n \geq 1 - (1 - \lambda \sigma_i^2)^n\,, \quad \text{for } i = 1, \ldots, N\,.$$

After some straightforward manipulations, we get that the above condition is satisfied if, for instance,

$$n \geq -\log(\lambda \sigma_i^{2k} + 1)/\log(1 - \lambda \sigma_i^2)\,.$$

Regarding the last hypothesis (4.20) of Theorem 4.3, we exploit the following relation

$$\mathbf{x} - \mathbf{x}_n = P(I - \lambda A^T A)^n \mathbf{x} \,,$$

which is immediate if we substitute $A\mathbf{x}$ instead of $\mathbf{b}$ in the expression (6.7) of $\mathbf{x}_n$. Then, thanks to the bounds (5.7), we have

$$\lim_{n\to\infty} \frac{\langle (I - \lambda A^T A) P A^T \mathbf{b}, (I - \lambda A^T A)^n \mathbf{x} \rangle}{\langle P A^T \mathbf{b}, (I - \lambda A^T A)^n \mathbf{x} \rangle} \neq 1 \,.$$

Therefore, under these assumptions, convergence is guaranteed. In the next section we test the performance of this choice of $\mathbf{y}$ .

## 6.4 Numerical experiments

In this section we perform several experiments that illustrate the gain of using STEA for extrapolating mainly Landweber and Cimmino methods. Comparisons with other iterative methods and acceleration techniques are also performed. We stress that, in order to illustrate the behavior of the methods, in most of our numerical experiments we consider a fixed number of iterations in Algorithms 1 and 2. We recall that STEA allows us to improve the numerical stability (with respect to TEA), since the potential singularities in (4.16) can be overcome by using the particular rules (see Section 1.4). In our experiments the gain from handling singularities is not significant; for this reason, in order to simplify the presentation and the discussion of the results, we always keep the number of digits $p = 12$, so that no singularities are met (cf. eq. (2.4)). Note that in all the numerical experiments we have used formula (4.16).

Regarding the relaxation parameter in (5.5), for Cimmino method we consider the options

$$\bar{\lambda} = 2 \quad \text{or} \quad \lambda_{\text{AIR}} = \frac{1.9}{\|A^T D A\|} \,. \tag{6.8}$$

The first one was originally adopted by Cimmino himself [27], while the second one is the default value employed in *AIR Tools*, Version 1.2. For Landweber method we consider the options

$$\bar{\lambda} = \frac{1}{\parallel A \parallel_1 \parallel A \parallel_\infty} \quad \text{or} \quad \lambda_{\text{AIR}} = \frac{1.9}{\|A\|^2} \,. \tag{6.9}$$

The first one is proposed in [5], while the second one is the default value employed in *AIR Tools*, Version 1.2. As a matter of notation, in all the following tables “error$_{\text{opt}}$”

denotes the minimum relative error, obtained at iteration "$\mathrm{it}_{\mathrm{opt}}$" of an iterative method. "$T$" denotes the time required for performing "$\mathrm{it}_{\mathrm{opt}}$" iterations, "tot.T" denotes the time required for performing a fixed number of iterations, and "avg.T" is the average time per iteration. All the computational times are measured in seconds. All the tests are performed with Matlab R2013a.

**Example 1.** We consider the `seismictomo` test problem whose data are shown in Figure 6.1. First we try to accelerate all the methods included in the *AIR Tools* package by using Algorithm 1 with $\mathbf{y} = \mathbf{1}$. We stress that Algorithm 1 is easily adapted for ART methods. In Table 6.1 we report the minimum relative error among those obtained during the first 10 iterations.

| | Without acceleration | | | STEA acceleration | | |
|---|---|---|---|---|---|---|
| | $\mathrm{it}_{\mathrm{opt}}$ | $\mathrm{error}_{\mathrm{opt}}$ | $T$ | $\mathrm{it}_{\mathrm{opt}}$ | $\mathrm{error}_{\mathrm{opt}}$ | $T$ |
| Landweber (with $\lambda_{\mathrm{AIR}}$) | 10 | 0.2996 | 2.82 | 9 | 0.2077 | 2.95 |
| Cimmino (with $\lambda_{\mathrm{AIR}}$) | 10 | 0.2764 | 1.67 | 10 | 0.2001 | 1.97 |
| CAV | 10 | 0.2863 | 2.41 | 10 | 0.1826 | 2.63 |
| DROP | 10 | 0.3613 | 1.60 | 10 | 0.2352 | 1.81 |
| SART | 10 | 0.3310 | 0.57 | 10 | 0.2246 | 0.80 |
| Kaczmarz | 6 | 0.6288 | 9.94 | 5 | 0.5959 | 9.97 |
| Symmetric Kaczmarz | 2 | 0.9344 | 6.51 | 2 | 0.9361 | 6.73 |
| Randomized Kaczmarz | 1 | 0.3738 | 50.19 | 3 | 0.3589 | 50.87 |

Table 6.1: `seismictomo` test problem. Minimum relative error and computational time averaged over 100 different runs of SIRT and ART methods and their extrapolated versions (by Algorithm 1).

Speaking for the class of SIRT methods, we observe that the use of STEA results in a smaller relative error. In particular, for CAV method the error of the accelerated method is 36.22% smaller than the error of the unaccelerated method. Note also that in the case of Landweber method we not only have a better accuracy, but also a smaller number of iterations. The same holds for Kaczmarz method, while STEA applied to the symmetric Kaczmarz diverges immediately. Regarding randomized Kaczmarz, the result may vary due to the randomness of the original method. Here, we present an interesting example where STEA succeeds in approximating the solution better than the iterative method, which diverges after the 1st iteration. In all the cases, when applying STEA, we observe a slight increase in the total computational time, which is due to the extrapolation procedure. However, the extra computational cost is negligible, and this result agrees with Remark 6.1.

We now focus on Cimmino method. Despite the fact that the value $\lambda_{\mathrm{AIR}}$ in (6.8) results in a faster convergence rate, we will attempt to accelerate the classical Cimmino method (i.e., with $\lambda = \bar{\lambda} = 2$), since estimating $\|A^T DA\|$ increases the computational cost. We apply STEA to Cimmino method, trying both Algorithm 1 with $\mathbf{y} = \mathbf{1}$ and Algorithm 2 with several options for the vector $\mathbf{y}$, and different combinations of cycles and inner iterations (in the following, if $m$ cycles of $\ell$ iterations are considered, we will denote it by $m \times \ell$ iterations). The relative error of Cimmino method itself after 100 iterations is 0.5144, while Algorithm 1 with $\mathbf{y} = \mathbf{1}$ gives a relative error equal to 0.2244 (i.e., 56.36% smaller). The relative errors resulting from Algorithm 2 are displayed in Table 6.2. We stress that, when writing $\mathbf{y} = \mathbf{r}_0^{(j)}$ for an overdetermined problem as this one, we only consider the first $N$ components of $\mathbf{r}_0^{(j)}$. Also, since in our tests we always use $\mathbf{x}_0^{(0)} = \mathbf{0}$, when implementing Algorithm 2 with $\mathbf{y} = \mathbf{x}_0^{(j)}$, we set $\mathbf{y} = \mathbf{b}$ for the first cycle.

| | $4 \times 25$ | $2 \times 50$ | $5 \times 20$ | $20 \times 5$ | $25 \times 4$ |
|---|---|---|---|---|---|
| $\mathbf{y} = \mathbf{1}$ | 0.2368 | 0.2557 | 90.6303 | 0.4861 | 0.6438 |
| $\mathbf{y} = \mathbf{r}_0^{(j)}$ | 0.1567 | 0.2244 | 0.1956 | 0.3423 | 0.3659 |
| $\mathbf{y} = \mathbf{x}_0^{(j)}$ | 0.2243 | 0.1843 | 0.2741 | 0.4233 | 0.3169 |
| $\mathbf{y} = -A^T\mathbf{b}$ | 0.1979 | 0.2425 | 0.3685 | 0.4395 | 0.6308 |

Table 6.2: `seismictomo` test problem. Relative errors of Cimmino accelerated by STEA following the restarted technique with various choices for the vector $\mathbf{y}$.

Analyzing the data reported in Table 6.2 we reach the conclusion that $\mathbf{y} = \mathbf{1}$ should be avoided when using Algorithm 2, since it cannot guarantee the convergence. The choices $\mathbf{y} = \mathbf{r}_0^{(j)}$ and $\mathbf{y} = \mathbf{x}_0^{(j)}$ both perform well. The choice $\mathbf{y} = -A^T\mathbf{b}$ is competitive to the previous ones for cycles with 25 or 50 iterations, while when we consider fewer inner iterations the relative errors increase. The same phenomenon has been analyzed in

Section 6.3 for STEA applied to the non negatively projected Ladweber method (5.24). The best result is anyway obtained when we apply 4 × 25 iterations of the Algorithm 2 with $\mathbf{y} = \mathbf{r}_0^{(j)}$. In Figure 6.5, we may compare the quality of the reconstructions brought by the iterative method itself, with acceleration (Algorithm 1 with $\mathbf{y} = \mathbf{1}$), and using the restarted technique (Algorithm 2 with $\mathbf{y} = \mathbf{r}_0^{(j)}$).

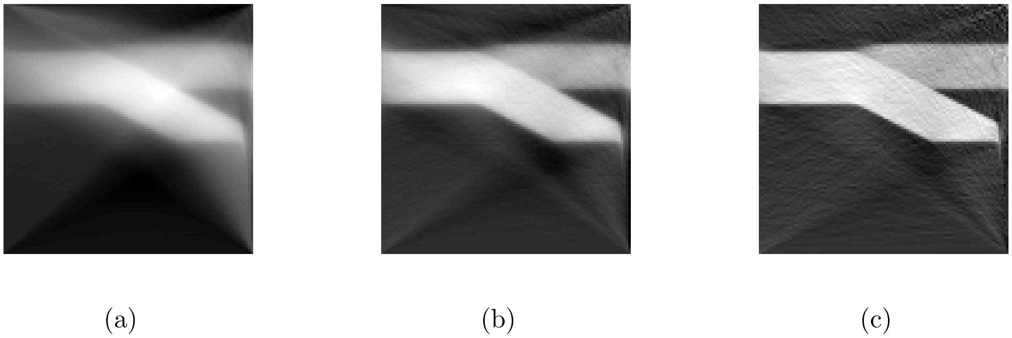
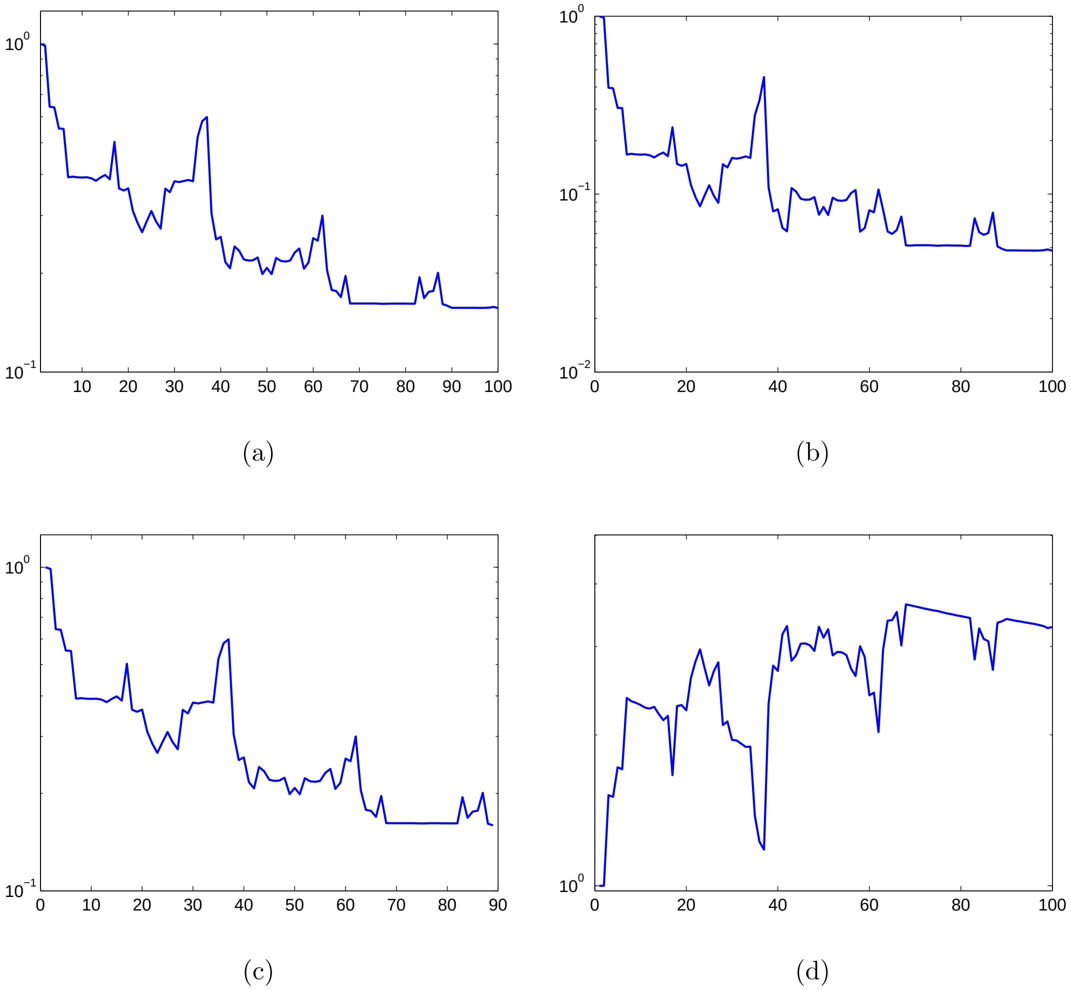


Figure 6.5: Reconstructions of the `seismictomo` image. (a) Cimmino method, 100 iterations. (b) Algorithm 1, 100 iterations. (c) Algorithm 2, 4 × 25 iterations.

Let us analyze more deeply this last case of Algorithm 2. The error history in Figure 6.6a shows the fast convergence of the Cimmino method when STEA is applied. We observe some peaks in random iteration steps, which we may overcome by monitoring the residual, whose graph in Figure 6.6b closely follows the graph of the error. The residual can also be used for defining a stopping criterion. In particular, we may apply the discrepancy principle [41], i.e., if we assume that a good estimate of the noise level $\delta$ is available, then we can stop as soon as the residual lies below the noise level. As we see in Figure 6.6c, for this problem the discrepancy principle gives slightly different results by terminating the process after 89 total iterations. An alternative stopping criterion (originally proposed in [17]) may be obtained by monitoring the ratios $\|\mathbf{z}_{n+1} - \mathbf{z}_n\|/\|\mathbf{x}_{n+1} - \mathbf{x}_n\|$, where $\mathbf{z}_n$ denotes the vectors computed by the extrapolation procedure (see Algorithm 2). Indeed, Figure 6.6d looks like the "complementary" of Figure 6.6a, since when the values of the ratio grow (decrease) then the values of the error decrease (grow).



(a)

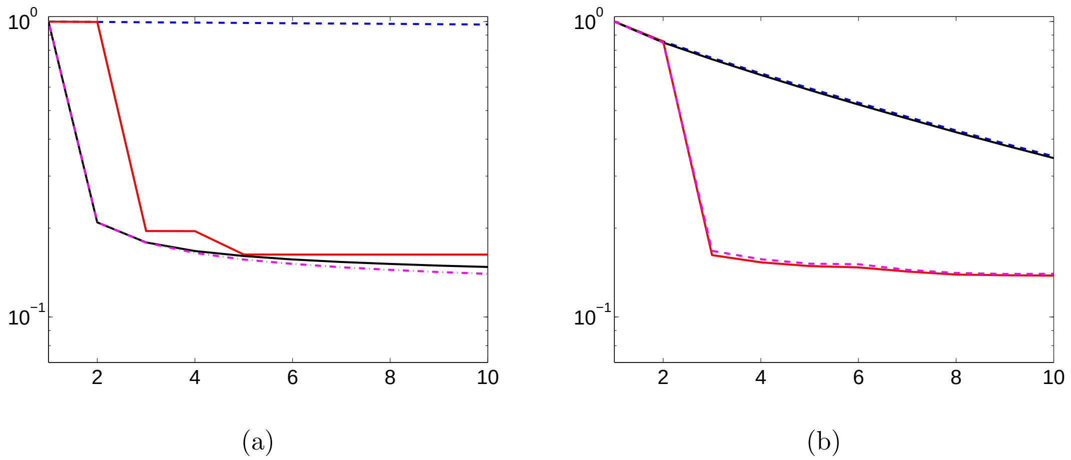
(b)

(c)

(d)

Figure 6.6: `seismictomo` test problem, Algorithm 2 for Cimmino method 4 × 25 iterations. (a) Relative error history. (b) Relative residual history. (c) Relative error history when the discrepancy principle is used. (d) Ratios $\|\mathbf{z}_{n+1} - \mathbf{z}_n\| / \|\mathbf{x}_{n+1} - \mathbf{x}_n\|$.

**Example 2.** We consider the `cameraman` test problem. In Figure 6.7 we plot the graphs of the relative errors obtained by applying Landweber and Cimmino methods with different relaxation parameters, together with their extrapolated versions (by Algorithm 1). In particular, referring to (6.8) and (6.9), in Figure 6.7a we take the values $\bar{\lambda}$, while in Figure 6.7b we take the values $\lambda_{\text{AIR}}$. We only perform 10 iterations of both methods, and we take $\mathbf{y} = \mathbf{1}$. The performance of STEA is especially good for Cimmino method with parameter $\lambda = 2$: the error is 83.36% smaller, and the quality of the reconstructions is similar to the one obtained by Cimmino method with $\lambda_{\text{AIR}}$, at a lower computational cost (since no preprocessing for computing the relaxation parameter has to be performed). While the computational time for the former method is around 0.15 seconds, the latter takes about 13 seconds.



(a)

(b)

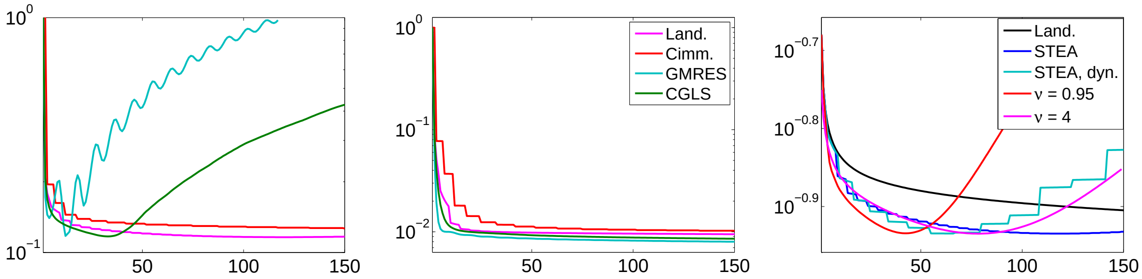
Figure 6.7: `cameraman` test problem. Relative errors using Cimmino (dashed lines) and Landweber (dashed-dotted line) methods, and their extrapolations (red and black solid lines, respectively). (a) $\bar{\lambda}$ defined in (6.8) and (6.9) is employed. (b) $\lambda_{\text{AIR}}$ defined in (6.8) and (6.9) is employed.

From now on, we only consider the restarted extrapolated versions of Cimmino and Landweber methods (Algorithm 2), with $\mathbf{y} = \mathbf{r}_0^{(j)}$ and $20 \times 5$ iterations. As relaxation parameters we take $\bar{\lambda}$ defined in (6.8) and (6.9) for Cimmino and Landwber methods, respectively. We also consider a "dynamic" version of Algorithm 2, where the number of iterations of each inner cycle is increased by one at each restart. The reason behind this modification is that, as the number of total iterations increases, the decrease of the relative error tends to slow down: in this situation it is meaningful and beneficial to extrapolate on more iterates.

In Figure 6.8 we compare the performance of different iterative regularizing methods, including some Krylov subspace methods and $\nu$-methods. We clearly see that the extrapolated Cimmino and Landweber methods eventually deliver approximations of the same quality as CGLS method [57] and some $\nu$-methods, but they exhibit a much more stable behavior. In particular, the extrapolated Cimmino and Landweber methods are not much affected by semiconvergence. This is expected, since Cimmino and Landweber methods display a very slow convergence. When considering Algorithm 2 applied to Cimmino and Landweber, the decrease of the relative errors within an inner iteration cycle is very slow and the overall scheme is quite stable. We also note that the extrapolated methods outperform GMRES method [57]. Probably, the GMRES performs badly for this test problem because the PSF is highly unsymmetric (see the analysis in [33]). Moreover, while the residuals associated to Krylov subspace methods (i.e., CGLS and GMRES) rapidly stabilize and are almost constant even when the relative errors worsen, the residuals associated to the extrapolated vector mimic quite well the behavior of the relative errors. For this reason, it is easy to detect the deterioration of the solution by looking at the residuals, and we should expect a stopping criterion based on the residual (such as the discrepancy principle), to be successful when applied to the extrapolated vectors.

In Table 6.3 we report the minimum relative error achieved by all the methods taken into account, together with the performed iterations and some timings. We must remark that the computational time of STEA is sensibly higher than the computational time of the other methods: the reason behind this drawback is that STEA as described in Algorithm 2 is the only method based on restarts, and MATLAB is notoriously slow when dealing with cycles. We also note that the performance of the $\nu$-methods is dependent on the parameter $\nu$. The choice of $\nu$ theoretically depends on the regularity of the solution: in practice, a too small $\nu$ can lead to instability, while a too big $\nu$ is not very effective in speeding up the convergence. STEA method depends on the choice of $\mathbf{y}$; the remarks in Section 6.3, and the numerical experiments performed so far have shown that the choice $\mathbf{y} = \mathbf{r}_0^{(j)}$ is reliable.

We conclude this example by showing in Figure 6.9 some restored images.



(a) (b) (c)

Figure 6.8: `cameraman` test problem. Comparisons of different iterative methods (frames (a) and (b) share the same legend, and the legend items "Land." and "Cimm." stand for the extrapolated versions of the Landweber and Cimmino methods, respectively). (a) History of the relative errors. (b) History of the relative residuals. (c) History of the relative errors of the $\nu$-methods with $\nu = 0.95$, 4, and the unaccelerated and dynamic STEA versions of the Landweber method.

**Example 3.** We consider the `satellite` test problem, with different noise levels $\delta$, and we test the performance of the projected Landweber and SD methods (the latter is also called projected gradient method). As said in Section 6.3, these methods incorporate some constraints on the approximate solutions; in our case, we consider nonnegativity and conservation of volume (5.25). In Figure 6.10 we apply Algorithm 2 to both the projected Landweber and SD methods. We consider two versions of Algorithm 2. For some tests, we take a fixed number of iterations. In particular, we perform $10 \times 10$ iterations of each method. For some other tests, we consider a stopping criterion based on the discrepancy principle. Namely, we stop as soon as the relative residual lies below the noise level $\delta$, or as soon as a maximum number $\ell$ of iterations per cycle is reached (in our experiments, $\ell = 20$). We also compare different choices of the vector $\mathbf{y}$. It is particularly evident that the restarted STEA algorithm is very effective with Landweber-type methods. Indeed, for the choices $\mathbf{y} = \mathbf{r}_0^{(j)}$ and $\mathbf{y} = -A^T\mathbf{r}_0^{(j)}$, the behavior of the relative error is quite stable (provided that the iterations are suitably stopped), and the quality of the restorations is very good: this agrees with the analysis performed in Section 6.3. In particular, the extrapolated projected Landweber method outperforms the extrapolated projected SD method.

In Figure 6.11 we set under comparison the Projected Landweber method with $\delta = 10^{-1}$,

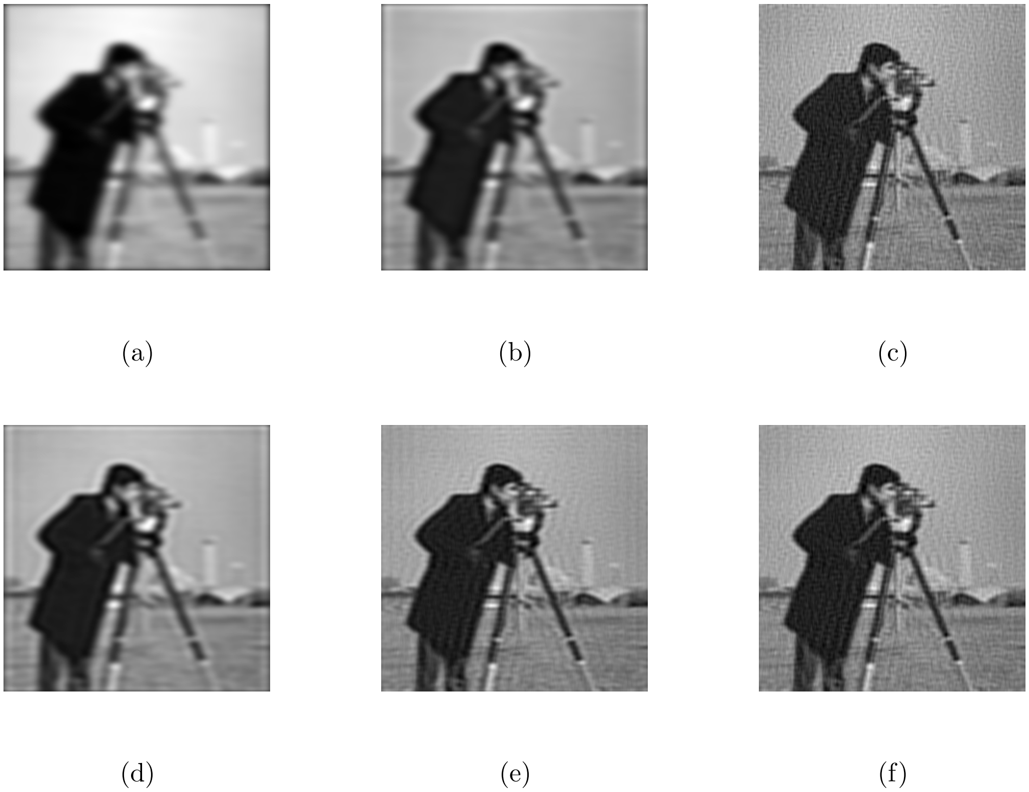
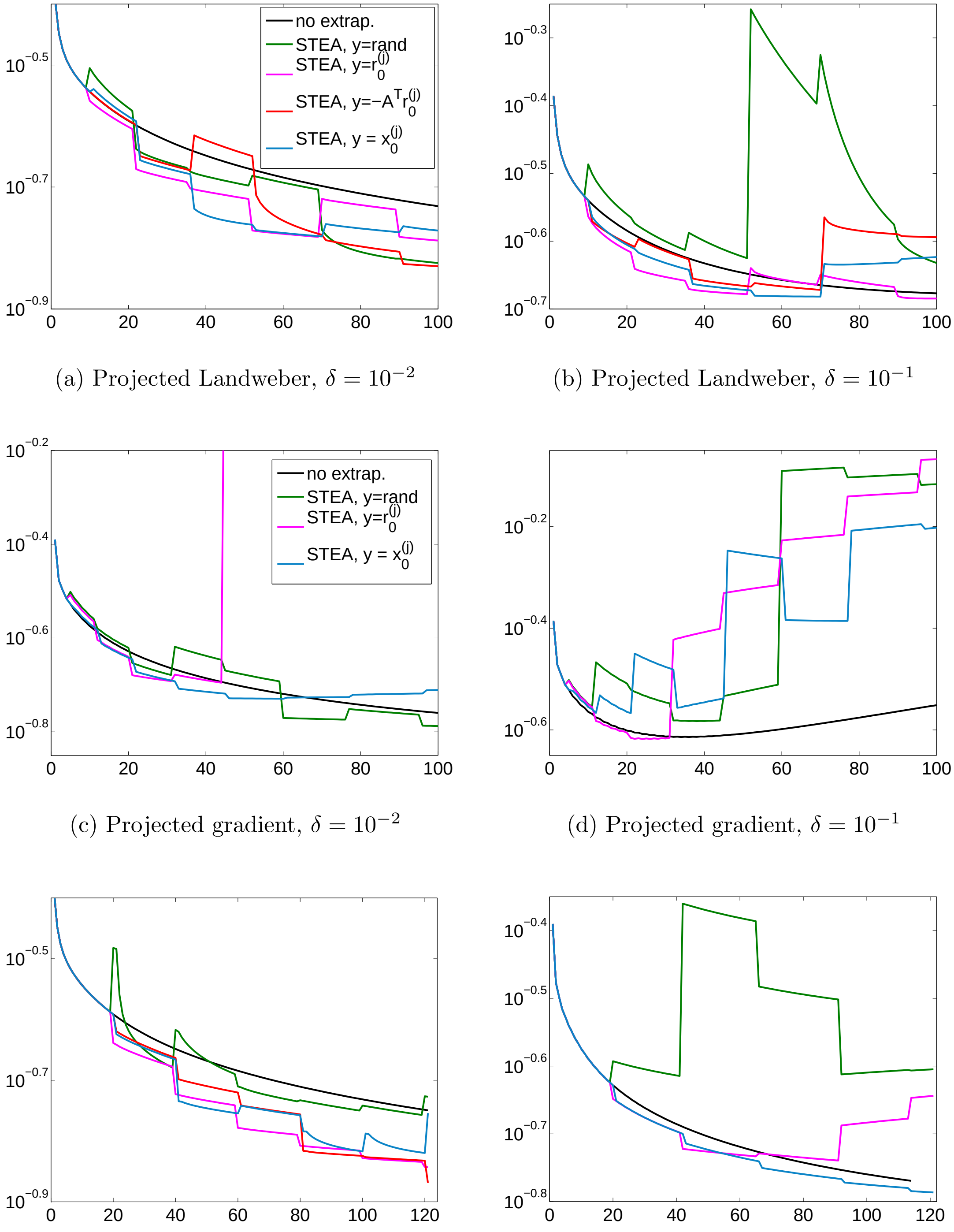


Figure 6.9: Restorations of the `cameraman` image. (a) Cimmino method, 150 iterations. (b) Algorithm 1 for Cimmino method, 150 iterations. (c) Algorithm 2 for Cimmino method, 30 × 5 iterations. (d) Landweber method, 150 iterations. (e) Algorithm 1 for Landweber method, 150 iterations. (f) Algorithm 2 for Landweber method, 30 × 5 iterations.



(a) Projected Landweber, $\delta = 10^{-2}$

(b) Projected Landweber, $\delta = 10^{-1}$

(c) Projected gradient, $\delta = 10^{-2}$

(d) Projected gradient, $\delta = 10^{-1}$

(e) Projected Landweber, $\delta = 10^{-2}$

(f) Projected gradient, $\delta = 10^{-2}$

Figure 6.10: `satellite` test problem. History of the relative errors for various projected methods and their restarted extrapolated versions. For (a)-(d) 10 × 10 iterations have been performed. For (e)-(f) a stopping criterion based on the discrepancy principle has been used.

| | $\text{error}_{\text{opt}}$ | $\text{it}_{\text{opt}}$ | tot.T | avg.T |
|---|---|---|---|---|
| Landweber | 0.1245 | 150 | 9.38 | 0.06 |
| Cimmino | 0.6865 | 150 | 62.97 | 0.42 |
| Cimm. STEA | 0.3846 | 12 | 4.53 | 0.37 |
| Cimm. dynamic STEA | 0.1266 | 150 | 63.26 | 0.42 |
| Land. STEA | 0.1162 | 119 | 53.37 | 0.45 |
| Land. dynamic STEA | 0.1163 | 67 | 25.64 | 0.38 |
| $\nu = 0.50$ | 0.1172 | 33 | 1.69 | 0.05 |
| $\nu = 0.95$ | 0.1163 | 43 | 2.10 | 0.05 |
| $\nu = 1.05$ | 0.1163 | 44 | 2.88 | 0.06 |
| $\nu = 4.00$ | 0.1162 | 79 | 5.57 | 0.07 |
| CGLS | 0.1171 | 33 | 1.45 | 0.04 |
| GMRES | 0.1176 | 12 | 0.41 | 0.03 |

Table 6.3: `cameraman` test problem. Minimum relative error, attained at the iteration $\text{it}_{\text{opt}}$.

extrapolated by restarted STEA with $\mathbf{y} = \mathbf{r}_0^{(j)}$, with an accelerated version of the Richardson-Lucy algorithm [8], as implemented in the MATLAB function `deconvlucy`; 5 cycles of 10 iterations are performed, taking as new initial guess at the beginning of each cycle the approximation computed at the end of the previous cycle. We can note a rapid decrease of the error during the first cycle of iteration, but a sudden deterioration of the approximations during the later cycles. In Figure 6.12 we display the restored images obtained by different methods that enforce nonnegativity constraints. The noise level is $\delta = 10^{-1}$, and we take the optimal number of (total) iterations, i.e., the ones that minimizes the relative error. All the comparisons are summarized in Table 6.4.

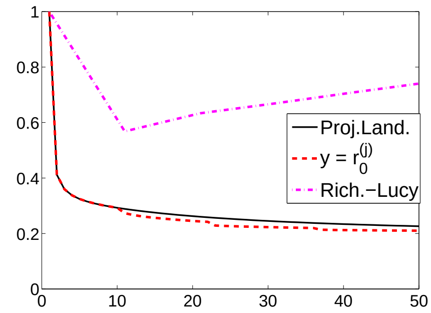
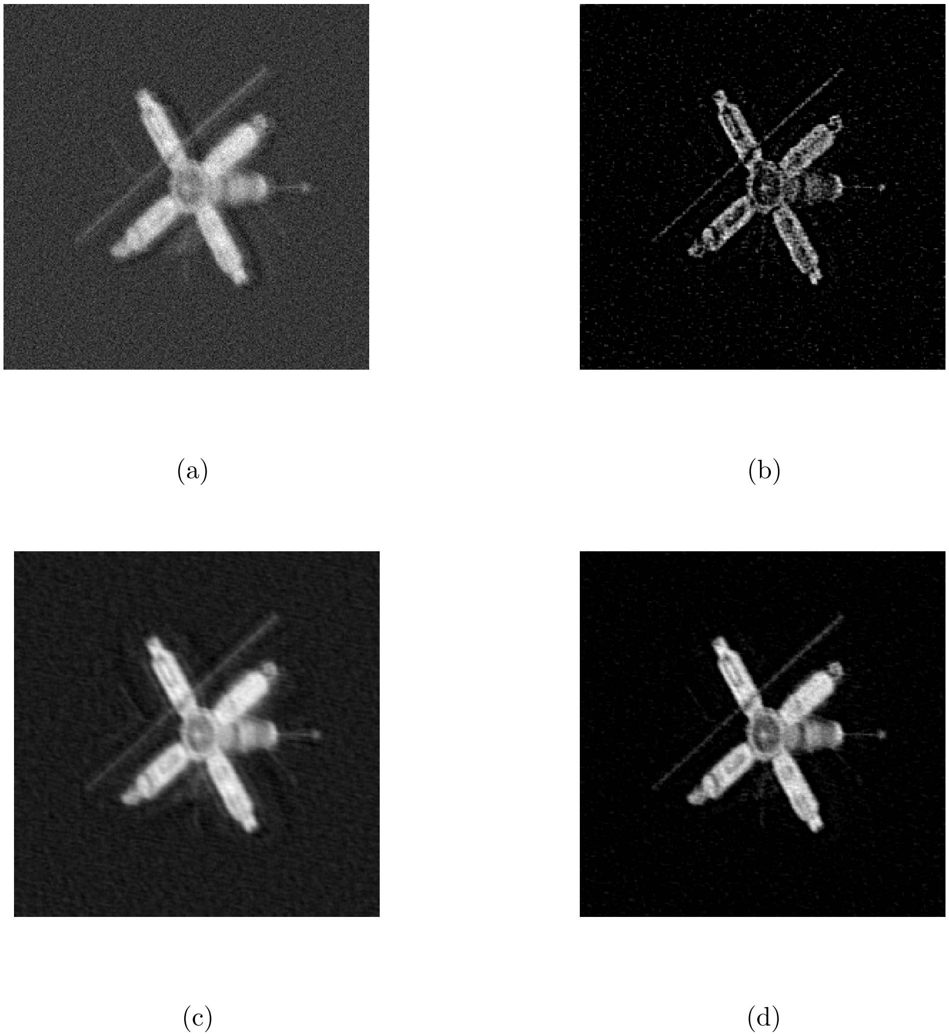


Figure 6.11: `satellite` test problem. Projected Landweber method with $\delta = 10^{-1}$, restarted STEA with $\mathbf{y} = \mathbf{r}_0^{(j)}$, and accelerated Richardson-Lucy algorithm.

| | error$_{\text{opt}}$ | it$_{\text{opt}}$ | tot.T | avg.T |
|---|---|---|---|---|
| Proj. Landweber | 0.2103 | 100 | 5.42 | 0.05 |
| STEA, $\mathbf{y} =$`rand` | 0.2092 | 100 | 7.33 | 0.07 |
| STEA, $\mathbf{y} = \mathbf{x}_0^{(j)}$ | 0.2081 | 70 | 6.42 | 0.06 |
| STEA, $\mathbf{y} = \mathbf{r}_0^{(j)}$ | 0.2067 | 98 | 7.99 | 0.08 |
| Richardson-Lucy | 0.5681 | 10 | 0.72 | 0.07 |

Table 6.4: `satellite` test problem, with $\delta = 10^{-1}$. We compare the results of the methods for constrained minimization, including the accelerated Richardson-Lucy algorithm. Minimum relative error, attained at the iteration it$_{\text{opt}}$.



(a)

(b)

(c)

(d)

Figure 6.12: Restorations of the `satellite` image. (a) Algorithm 2 with $\mathbf{y} = \mathbf{r}_0^{(j)}$ for Landweber method, 10 × 10 iterations. (b) Algorithm 2 with $\mathbf{y} = \mathbf{r}_0^{(j)}$ for projected Landweber method, 30 × 10 iterations. (c) Accelerated Richardson-Lucy method, 10 iterations. (d) Algorithm 2 with $\mathbf{y} = \mathbf{r}_0^{(j)}$ for projected gradient method, 20 × 10 iterations.

# Conclusion and Future work

In this thesis we discussed several aspects of scalar and vector extrapolation. After revising the most significant scalar transformations, we focused on the $\varepsilon$-algorithm and its particular rules. More precisely, we revised the particular rules proposed by Cordellier who extended Wynn's particular rules in the case of an arbitrary number of equal elements in the $\varepsilon$-array. The contribution of the author was the improvement of Cordellier's algorithm, overcoming its limitations, thus making it more efficient. As a future work, it could be interesting to attempt an extension to the vector case.

Another scalar transformation that gained our interest was $\Delta^2$ process for which we proposed some new generalizations. In particular, we introduced three new transformations which accelerate the convergence of sequences whose kernel contain that of $\Delta^2$ process. We studied extensively the transformation that performed better in the numerical tests. We analysed it theoretically, proving convergence and acceleration properties, under certain hypotheses. The numerical experiments included comparison among the proposed transformations and comparison of our best transformation with several well known transformations, using both convergent and divergent test sequences. It is remarkable that the numerical results showed that some of the conditions we assumed are not necessary. Therefore, as a future project it would be interesting to look for less restrictive assumptions.

One chapter of the thesis was dedicated to the revision of some well known iterative regularization methods, commonly used for the solution of a linear system of equations. Cimmino method with its interesting properties intrigued the author who studied it extensively, proposed a general formulation and a new variant, which however was proved to be competitive only in a special case.

Regarding the extrapolation methods for vector sequences, we only used the recently proposed simplified topological $\varepsilon$-algorithm, which we applied on several iterative regularization methods for image reconstruction and restoration problems. We mainly used Landweber and Cimmino methods giving algorithmic details and performing several numerical experiments and comparisons with other methods. Furthermore, we studied the acceleration properties of STEA applied to the nonnegatively projected Landweber method, and for a specific choice of the vector $\mathbf{y}$. We stress that the choice of the vector $\mathbf{y}$ is an important issue that affects the performance of the simplified topological $\varepsilon$-algorithm. Nevertheless, a theoretical study has never been presented. What we managed to do is to give an insight into the choice of the vector $\mathbf{y}$ when using STEA to accelerate Landweber method. As a future work, it would be interesting to extend the use of extrapolation in image reconstruction and restoration problems, by trying other known extrapolation methods, i.e., vector $\varepsilon$-algorithm (VEA), Minimal Polynomial Extrapolation (MPE), Reduced Rank Extrapolation (RRE), Modified Minimal Polynomial Extrapolation (MMPE), or even create new ones based on the special characteristics of the inverse problems at hand.

# Bibliography

[1] J. Abdalkhani, D. Levin, *On the choice of $\beta$ in the u-transformation for convergence acceleration*, Numer. Algorithms, 70 (2015) 205-213.

[2] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards, Washington, D. C., 1972.

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

[4] M. Benzi, *Gianfranco Cimmino's contributions to Numerical Mathematics*, Atti del Seminario di Analisi Matematica, Dipartimento di Matematica dell'Università di Bologna. Volume Speciale: Ciclo di Conferenze in Memoria di Gianfranco Cimmino, Marzo-Aprile 2004, Tecnoprint, Bologna (2005) 87-109.

[5] S. Berisha, J.G. Nagy, *Iterative image restoration*, in: R. Chellappa, S. Theodoridis, eds., Academic Press Library in Signal Processing, vol. 4, (7), pp. 193-243, Elsevier, 2014.

[6] R. Bertelle, M.R. Russo, *An approach to the Gummel map by vector extrapolation methods*, Numer. Algorithms, 45 (2007) 331-343.

[7] M. Bertero, P. Boccacci, *Introduction to Inverse Problems in Imaging*, Institute of Physics Publishing, London, 1998.

[8] D.S.C. Biggs, M. Andrews, *Acceleration of iterative image restoration algorithms*, Appl. Opt., 36-8 (1997) 1766-1775.

[9] Å. Björck, T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT 19 (1979) 145-163.

[10] S. Bonettini, R. Zanella, L. Zanni, *A scaled gradient projection method for constrained image deblurring*, Inverse Prob., 25 (2009).

[11] F. Bornemann, D. Laurie, S. Wagon, J. Waldvogel, *The SIAM 100-Digit Challenge: A Study in High-Accuracy Numerical Computing*, Society for Industrial and Applied Mathematics, Philadelphia, 2004.

[12] H. Brakhage, On ill-posed problems and the method of conjugate gradient, in: H. W. Engl, C. W. Groetsch, eds., Inverse and ill-posed problems, Academic press, Boston, New York, London (1987) 165-175.

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

[14] C. Brezinski, *Padé-type Approximation and General Orthogonal Polynomials*, Birkhäuser-Verlag, Basel, 1980.

[15] C. Brezinski, *Convergence acceleration during the 20th century*, J. Comput. Appl. Math, 122 (2000) 1-21.

[16] C. Brezinski, M. Redivo-Zaglia, *Construction of extrapolation processes*, Appl. Numer. Math., 8 (1991) 11-23.

[17] C. Brezinski, M. Redivo-Zaglia, *Convergence acceleration of Kaczmarz's method*, J. Eng. Math. 93 (2015) 3-19.

[18] C. Brezinski, M. Redivo-Zaglia, *Extrapolation Methods. Theory and Practice*, Studies in Computational Mathematics, North-Holland, Amsterdam, 1991.

[19] C. Brezinski, M. Redivo-Zaglia, Generalizations of Aitken's process for accelerating the convergence of sequences, Comput. Appl. Math., 26 (2007) 171-189.

[20] C. Brezinski, M. Redivo-Zaglia, *A primer on convergence acceleration methods and their applications*, Notes of the doctoral course Extrapolation methods and their applications, Padua, 2012.

[21] C. Brezinski, M. Redivo-Zaglia, *The simplified topological $\varepsilon$-algorithms for accelerating sequences in a vector space*, SIAM J. Sci. Comput., 36 (2014) A2227-A2247.

[22] C. Brezinski, M. Redivo-Zaglia, *The simplified topological $\varepsilon$-algorithms: software and applications*, in preparation.

[23] D. Buoso, A. Karapiperi, S. Pozza, *Generalizations of Aitken's process for a certain class of sequences*, Appl. Numer. Math., 90C (2015) 38-54.

[24] Y. Censor, T. Elfving, *Block-iterative algorithms with diagonally scaled oblique projections for the linear feasibility problem*, SIAM J. Matrix Anal. Appl., 24 (2002) 40-58.

[25] Y. Censor, T. Elfving, G.T. Herman, T. Nikazad, *On diagonally-relaxed orthogonal projection methods*, SIAM J. Sci. Comput., 30 (2008) 473-504.

[26] Y. Censor, D. Gordon, R. Gordon, *Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems*, Parallel Comput., 27 (2001) 777-808.

[27] C. Cimmino, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, La Ricerca Scientifica II, 9 (1938) 326-333.

[28] F. Cordellier, *Démonstration algébrique de l'extension de l'identité de Wynn aux tables de Padé non normales*, in: L. Wuytack, ed., Padé Approximation and its Applications, volume 765 of LNM, pp. 36-60, Berlin, 1979. Springer-Verlag.

[29] F. Cordellier, *Interpolation rationnelle et autres questions: Aspects algorithmiques et numériques*, Thèse d'État, Université de Lille 1, 1989.

[30] J.P. Delahaye, *Sequence Transformations*, Springer-Verlag, Berlin, 1988.

[31] J.P. Delahaye, B. Germain-Bonne, *Résultats négatifs en accélération de la convergence*, Numer. Math., 35 (1980) 443-457.

[32] J.P. Delahaye, B. Germain-Bonne, *The set of logarithmically convergent sequences cannot be accelerated*, SIAM J. Numer. Anal., 19 (1982) 840-844.

[33] M. Donatelli, D. Martin, L. Reichel, Arnoldi methods for image deblurring with anti-reflective boundary conditions. Appl. Math. Comput., 253 (2015) 135-150.

[34] H.W. Engl, M. Hanke, A. Neubauer, *Regularization of Inverse Problems*, Kluwer Academic Publishers, Dordrecht, 2000.

[35] S. Gazzola, A. Karapiperi, *Image reconstruction and restoration using the simplified topological $\varepsilon$-algorithm*, Appl. Math. Comput., 274 (2016) 539-555.

[36] S. Gazzola, P. Novati, M.R. Russo, On Krylov projection methods and Tikhonov regularization. Electron. Trans. Numer. Anal., 44 (2015) 83-123.

[37] A. Gil, J. Segura, N.M. Temme, *Numerical Methods for Special Functions*, SIAM, Philadelphia, 2007.

[38] R. Gordon, R. Bender, G.T. Herman, *Algebraic reconstruction techniques for* 3 *dimensional electron microscopy and X-ray photograph*, J. Theoret. Biol. 29 (1970) 471-481.

[39] P.R. Graves-Morris, *Vector valued rational interpolants*, I. Numer. Math., 42 (1983) 331-348.

[40] M. Hanke, *Accelerated Landweber iterations for the solution of ill-posed equations*, Numer. Math., 60 (1991) 341-373.

[41] P.C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion.*, SIAM, Philadelphia (PA), 1998.

[42] P.C. Hansen, *Regularization Tools Version 3.0 for Matlab 5.2.* Numer. Algorithms, 20 (1999) 195-196.

[43] P.C. Hansen, J.G. Nagy, D.P. OLeary, *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, Philadelphia (PA), 2006.

[44] P.C. Hansen, M. Saxild-Hansen, *AIR Tools  A MATLAB package of algebraic iterative reconstruction methods*, J. Comput. Appl. Math., 236 (2012) 2167-2178.

[45] M. Jiang, G. Wang, *Convergence of the Simultaneous Algebraic Reconstruction Technique (SART)*, IEEE Trans. Image Process., 12 (2003) 957-961.

[46] S. Kaczmarz, *Angenäherte Auflösung von Systemen linearer Gleichungen*, Bull. Acad. Pol. Sci., 35 A (1937) 355-357. English translation: *Approximate solution of systems of linear equations.* Int. J. Control, 57 (1993) 1269-1271.

[47] L. Landweber, *An iteration formula for Fredholm integral equations of the first kind*, Am. J. Math., 73 (1951) 615-624.

[48] K.P. Lee, J.G. Nagy, *Steepest descent, CG, and iterative regularization of ill-posed problems*, BIT, 43 (2003) 1003-1017.

[49] D. Levin, *Development of non-linear transformations for improving convergence of sequences*, Int. J. Comput. Math., 3 B (1973) 371-388.

[50] J.B. McLeod, *A note on the $\varepsilon$-algorithm*, Computing, 7 (1971) 17-24.

[51] J.G. Nagy, K.M. Palmer, L. Perrone, *Iterative methods for image deblurring: A Matlab object oriented approach*, Numer. Algorithms, 36 (2004) 73-93. See also *RestoreTools: An object oriented Matlab package for image restoration*, (2002), http://www.mathcs.emory.edu/ nagy/RestoreTools.

[52] F. Natterer, *The Mathematics of Computerized Tomography*, SIAM, Philadelphia (PA), 2001.

[53] A. Nikola, S. Petra, C. Popa, C. Schnörr, *On a general extending and constraining procedure for linear iterative methods*, Int. J. Comput. Math. (2011) 1-23.

[54] F.W.J. Olver, D.W. Lozier, R.F. Boisvert, C.W. Clark, *NIST Handbook of Mathematical Functions*, Cambridge U. P., Cambridge, 2010.

[55] N. Osada, *The early history of convergence acceleration methods*, Numer. Algorithms, 60 (2012) 205-221.

[56] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 2007.

[57] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd Edition, SIAM, 2003.

[58] M. Saxild-Hansen, *AIR Tools  A MATLAB Package for Algebraic Iterative Reconstruction Techniques*, M.Sc. Thesis, DTU Informatics, Technical University of Denmark, 2010.

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

[60] A. Sidi, *Practical Extrapolation Methods*, Cambridge University Press, Cambridge, 2003.

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

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

[63] D.A. Smith, W.F. Ford, A. Sidi, *Correction to "Extrapolation methods for vector sequences"*, SIAM Rev., 30 (1988) 623-624.

[64] T.J. Stieltjes, *Recherches sur quelques séries semi-convergentes*, Ann. Sci. Ec. Norm. Super., 3 (1886) 201-258.

[65] T. Strohmer, R. Vershynin, *A randomized solver for linear systems with exponential convergence*, Lect. Notes Comput. Sci., 4110 (2006) 499-507.

[66] K. Tanabe, *Projection method for solving a singular system of linear equations*, Numer. Math., 17 (1971) 203-214.

[67] L.N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, 2013.

[68] R.R. Tucker, *The $\delta^2$-process and related topics*, Pac. J. Math., 22 (1967) 349-359.

[69] E.J. Weniger, *Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series*, Comput. Physics Reports, 10 (1989) 189-371.

[70] E.J. Weniger, *A rational approximant for the digamma function*, Numer. Algorithms, 33 (2003) 499-507.

[71] E.J. Weniger, *Further discussion of sequence transformation methods*, Subtopic "Related Resources" (R1) on the Numerical Recipes (Third Edition) Webnotes page http://www.nr.com/webnotes/ (2007).

[72] E.J. Weniger, *On the mathematical nature of Guseinov's rearranged one-range addition theorems for Slater-type functions*, J. Math. Chem., 50 (2012) 17-81.

[73] J. Wimp, *Sequence Transformations and Their Applications*, Academic Press, New York, 1981.

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

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

[76] P. Wynn, *Singular rules for certain non-linear algorithms*, BIT, 3 (1963) 175-195.