N° d'ordre : 41366

UNIVERSITÉ DE LILLE 1- SCIENCES ET TECHNOLOGIES

École doctorale: Sciences Pour l'Ingénieur n° 72

Laboratoire Paul Painlevé

T H È S E

Pour obtenir le grade de

Docteur de l'Université de Lille 1

En MATHÉMATIQUES APPLIQUÉES

Présentée par

EL-MOALLEM Rola

EXTRAPOLATION VECTORIELLE ET APPLICATIONS AUX MÉTHODES ITÉRATIVES POUR RÉSOUDRE DES ÉQUATIONS ALGÉBRIQUES DE RICCATI

Thèse dirigée par : **BELMEHDI Said** et **SADOK Hassane**

Soutenue le
Jeudi 12 Decembre , 2013

JURY

Rapporteurs :	SADKANE Miloud	-	Université Bretagne Occidentale, France
	VAN BAREL Marc	-	Université catholique de Louvain, Belgique
Directeurs de thèse :	BELMEHDI Saïd	-	Université de Lille 1, France
	SADOK Hassane	-	Université du Littoral Côte d'Opale, France
Examinateurs :	BECKERMANN Bernhard	-	Université de Lille 1, France
	LABAHN George	-	University of Waterloo, Canada
Président du jury :	CHEHAB Jean-Paul	-	Universite de Picardie Jules Verne, France

N° d'ordre : 41366

UNIVERSITY OF LILLE 1- SCIENCES AND TECHNOLOGIES

Doctoral school: Sciences Pour l'Ingénieur n° 72

Laboratoire Paul Painlevé

THESIS

To obtain the title of

PhD of Science

In APPLIED MATHEMATICS

Defended by

ROLA EL-MOALLEM

APPLICATION OF VECTOR EXTRAPOLATION ON ITERATIVE METHODS TO SOLVE ALGEBRAIC RICCATI EQUATIONS

Thesis Advisors:
Said BELMEHDI and **Hassane SADOK**

Defended on
Thursday December 12th, 2013

DISSERTATION COMMITTEE

Reviewers :	Miloud SADKANE	-	University of Western Brittany, France
	Marc VAN BAREL	-	Catholic University of Leuven, Belgium
Advisors :	Said BELMEHDI	-	University of Lille 1, France
	Hassane SADOK	-	University of the Littoral Opal Coast, France
Examinators :	Bernhard BECKERMANN	-	University of Lille 1, France
	George LABAHN	-	University of Waterloo, Canada
President :	Jean-Paul CHEHAB	-	Université de Picardie Jules Verne, France

This thesis was prepared at

Université de Lille 1- Sciences et Technologies
59655 Villeneuve d'Ascq Cedex
France
Tel. +33 (0) 3 20 43 43 43
Website: http://www.univ-lille1.fr

Laboratoire Paul Painlevé
Bâtiment M2, Cité Scientifique
59 655 Villeneuve d'Ascq Cédex
France
Tel : +33 (0)3 20 43 48 50
Fax : +33 (0)3 20 43 43 02
Website: http://labomath.univ-lille1.fr/
Email: webmaster@math.univ-lille1.fr

Keywords: Vector extrapolation, reduced rank extrapolation (RRE), minimal polynomial extrapolation (MPE), modified minimal polynomial extrapolation (MMPE), convergence acceleration, restarted algorithms, iterative methods, vector sequences, nonlinear system of equations, nonsymmetric algebraic Riccati equation, transport theory, minimal positive solution, Jacobian matrix, critical case.

Mots clés : Extrapolation vectorielle, reduced rank extrapolation (RRE), minimal polynomial extrapolation (MPE), modified minimal polynomial extrapolation (MMPE), accélération de la convergence, méthodes redémarrées, méthodes itératives, suite de vecteurs, systèmes non linéaires, équation de Riccati nonsymmetrique algébrique , théorie du transport, une solution minimale positive, une matrice Jacobienne, cas critique.

Abstract

In this thesis, we are interested in the study of polynomial extrapolation methods and their application as convergence accelerators on iterative methods to solve Algebraic Riccati equations arising in transport theory . In such applications, polynomial extrapolation methods succeed to accelerate the convergence of these iterative methods, even in the most critical region where the convergence turns to be extremely slow.

The advantage of these methods of extrapolation is that they use a sequence of vectors which is not necessarily convergent, or which converges very slowly to create a new sequence which can admit a quadratic convergence. Furthermore, the development of restarted (or cyclic) methods allows to limit the cost of computations and storage.

We search for the most efficient iterative methods used to solve such kind of Riccati equations. Convergence problems of these methods are examined and critical regions where the convergence turns to be very slow are located. Then, we apply polynomial extrapolation to these iterative methods to improve the convergence, especially in these regions.

An interpretation of the critical case which is the most challenging problem is made. In this case, the Jacobian matrix at the required solution is singular and quadratic convergence turns to linear. This problem can be overcome by applying a suitable shift technique in order to get rid of the singularity. The original equation is transformed into an equivalent Riccati equation where the singularity is removed while the matrix coefficients maintain the same structure as in the original equation. The nice feature of this transformation is that the new equation has the same solution as the original one although the new Jacobian matrix at the solution is nonsingular. Numerical experiments and comparisons which confirm the effectiveness of the new approaches are reported.

Résumé

Nous nous intéressons, dans cette thèse, à l'étude des méthodes d'extrapolation polynomiale ainsi qu'à leurs applications à l'accélération de méthodes itératives pour la résolution d'un cas particulier de l'équation algébrique de Riccati utilisée dans la théorie de transport. Pour ce type d'applications, l'extrapolation polynomiale permet d'accélérer la convergence des méthodes itératives et ceci même pour des cas critiques où la convergence devient extrêmement lente. L'avantage de ces méthodes d'extrapolation est qu'elles utilisent uniquement une suite de vecteurs qui n'est pas forcément convergente, ou qui converge très lentement pour créer une nouvelle suite qui converge plus vite et pouvant admettre une convergence quadratique. De plus, le développement de méthodes redémarrées (ou cycliques) permet de limiter le coût et le stockage.

Nous cherchons les méthodes itératives les plus efficaces pour la résolution de ce type d'équation de Riccati. Le problème de convergence de ces méthodes est examiné tout en identifiant les cas critiques correspondant à une convergence très lente. Ensuite, nous appliquons l'extrapolation polynomiale à ces méthodes afin d'améliorer leur convergence.

Une tâche importante relative à l'analyse du cas critique et son interprétation a été réalisée. Nous avons utilisé une technique de décalage « shift technique » afin d'éliminer le problème lié à la singularité de la matrice Jacobienne. En résumé, en transformant l'équation de départ avec une technique de « shift » nous évitons le problème de singularité pour la matrice Jacobienne. L'efficacité de l'approche proposée est illustrée à travers plusieurs exemples numériques.

Contents

List of Tables

List of Figures

General Introduction

An important problem that arises in different areas of science and engineering is that of finding or approximating limits of infinite sequences of vectors $\{x^{(k)}\}$, where the $x^{(k)}$ are N-vectors with N very large. Such sequences may result from iterative methods or perturbation techniques and in most cases may converge extremely slowly to their limits with a desired accuracy. Thus, to approximate their limits with reasonable accuracy, one must compute a large number of the terms of $\{x^{(k)}\}$, and this is generally costly. These limits can be approximated economically and with high accuracy by applying suitable extrapolation (or convergence acceleration) methods to a small number of terms of $\{x^{(k)}\}$.

This is the case, for example, when they result from the finite-difference or finite-element discretizations of continuum problems, where their rates of convergence become worse as the relevant mesh sizes get smaller. This requires the use of convergence acceleration methods. Vector extrapolation methods are techniques which can be applied to such vector sequences.These methods transform a sequence of vectors generated by some process to a new one so that it converges faster than the initial sequence. An example to these vector sequences is those which are obtained from iterative solution of linear and nonlinear systems of equations. The limits of these sequences are simply the required solutions of these systems.

In this thesis, we are interested in extrapolation methods. These methods can be scalar or vector ones. In the scalar case, the Richardson extrapolation and the Aitken's Δ^2-process are two popular representatives. For the vector case , these methods can be classified into two main categories: the polynomial methods and the ϵ-algorithms. We will be interested in the polynomial methods. There exists many polynomial extrapolation methods but the most popular methods among them are the minimal polynomial extrapolation (MPE) method of Cabay and Jackson [10], the reduced rank extrapolation (RRE) method of Eddy [14] and Mesina [48], and the modified minimal polynomial extrapolation (MMPE) method of Sidi et al. [69], Brezinski [9] and Pugachev [53]. These methods do not require an explicit knowledge of how the sequence is generated, and consequently can be directly applied for solving linear and nonlinear systems where the Jacobian of the function is not needed for those which are nonlinear. They are however more effective when they are applied to the resolution of systems of non linear equations.

This thesis is organized as follows:

Chapter 1 presents a glossary of symbols in addition to some notations and definitions which will be used throughout the thesis.

Chapter 2 deals with extrapolation methods: the theory of extrapolation, types (scalar and vector) with emphasizing on the vector extrapolation methods and, in particular, the polynomial methods. These methods, namely, the minimal polynomial extrapolation (MPE) and the reduced rank extrapolation (RRE) are timewise efficient and numerically stable convergence accelerators. Description of the algorithms of these methods and their applications are described. We will be interested later in the application of polynomial methods to nonlinear systems of equations.

Chapter 3 is devoted to a certain kind of Riccati equations, in particular the nonsymmetric algebraic Riccati equations (NARE), which arises in transport theory and which is our interest in this thesis. We will go over some iterative methods which have been proven to be efficient in solving these kinds of equations. We choose one method, namely the method of Y. Lin [44], and we propose a modification to it to accelerate it. Our modification is proved to be more efficient and outperforms the latter.

A combination of Chapter 2 and Chapter 3 leads to Chapter 4 where an application of vector extrapolation methods on the algebraic Riccati equations which arise in transport theory is conducted. Different ways of application are proposed followed by numerical experiments which shows the effectiveness of our approach.

Last Chapter of this thesis is an interpretation of the critical case where extremely slow convergence occurs. A simplification of the modified iterative scheme which was proposed in Chapter 3 is done. By this simplification, only half of the computational work will be needed. In the critical case, the Jacobian matrix at the required solution is singular and quadratic convergence turns to linear. This problem can be overcome by applying a suitable shift technique. Briefly speaking, the shift technique transforms NARE into another equation whose Jacobian matrix is nonsingular at the solution. The nice feature of this transformation is that the new equation has the same solution as the original one although the new Jacobian matrix at the solution is nonsingular.

Introduction Générale

Un problème important qui se pose dans différents domaines des sciences de l'ingénieur consiste à approcher la limite de suite de vecteurs $\{x^{(k)}\}$, où $x^{(k)}$ sont des N-vecteurs avec N très élevé. De telles suites peuvent résulter de méthodes itératives et dans beaucoup de cas convergent très lentement. Par conséquent, l'approximation de leur limite avec une précision raisonnable conduit à calculer un nombre important de termes de la suite $\{x^{(k)}\}$, ce qui est généralement très coûteux. Ces limites peuvent être approchées raisonnablement et avec une grande précision en appliquant des méthodes d'extrapolation convenables qui utilisent un petit nombre de termes de $\{x^{(k)}\}$.

Les méthodes d'extrapolation vectorielle sont très pertinentes car elles peuvent être appliquées afin de transformer une suite de vecteurs qui converge lentement en une suite convergeant rapidement.

Ces méthodes peuvent être scalaires ou vectorielles. Dans le cas scalaire, l'extrapolation de Richardson et le Δ^2 d'Aitken sont deux représentants connus et populaires. Dans le cas vectoriel, ces méthodes peuvent être classées en deux catégories principales : les méthodes polynomiales et les méthodes de type ϵ-algorithmes. Nous nous intéressons en particulier aux méthodes d'extrapolation polynomiales. Parmi les nombreuses méthodes d'extrapolation polynomiale, les plus populaires sont la méthode MPE (Minimal Polynomial Extrapolation) de Cabay et Jackson [10], la méthode RRE (Reduced Rank Extrapolation) d'Eddy [14] et Mesina [48] et la méthode MMPE (Modified Minimal Polynomial Extrapolation) de Sidi, Ford et Smith [69], Brezinski [9] et Pugachev [53]. Ces méthodes n'exigent pas une connaissance explicite sur la manière de générer la suite, et peuvent donc être appliquées à la résolution de systèmes linéaires ou non linéaires. Toutefois, elles sont plus efficaces lorsqu'elles sont appliquées à la résolution des systèmes d' équations non linéaires.

La thèse est divisée en plusieurs chapitres :
Les principales notations, symboles et définitions utilisés dans ce rapport sont données dans Chapitre 1.

Chapitre 2 présente les méthodes d'extrapolation, la théorie et les types des méthodes d'extrapolation (scalaire et vectorielle) tout en mettant l'accent sur les méthodes d'extrapolation vectorielles et en particulier les méthodes polynomiales. Ces méthodes comme en particulier la méthode MPE et la méthode RRE, sont des transformations d'accélération de la convergence très efficaces, peu couteuses et numériquement stables. Les algorithmes de ces méthodes et leurs applications sont décrits. Nous nous intéressons ensuite à l'application

des méthodes polynomiales aux systèmes d'équations non linéaires.

Chapitre 3 est consacrée à un certain type d'équations de Riccati, en particulier les équations de Riccati algébriques et non symétriques (NARE), que l'on obtient dans la théorie du transport et à laquelle on s'intéresse dans cette thèse. Nous présentons quelques méthodes itératives efficaces dans la résolution de ce type d'équations. Ensuite, nous nous focalisons sur la méthode de Y. Lin [44] et nous proposons une modification afin d'améliorer sa convergence.

Les deux dernières chapitres sont utilisées dans Chapitre 4 où une application de méthodes d'extrapolation vectorielles sur les équations Riccati algébriques est effectuée. Plusieurs types d'application sont proposés et illustrés à travers des résultats numériques qui montrent l'efficacité de l'approche utilisée.

Le dernier chapitre de cette thèse, Chapitre 5, concerne le cas critique où la convergence devient extrêmement lente. Nous proposons une simplification du schéma itératif modifié utilisée dans Chapitre 3. Grâce à cette simplification, le cout de calcul est divisé par deux. Finalement, nous définissons une technique de décalage appelée « shift technique » afin d'éliminer le problème lié à la singularité de la matrice Jacobienne ce qui rend la convergence linéaire plutôt que quadratique. En résumé, cette technique de « shift » transforme l'équation (NARE) en une autre dont la matrice jacobienne est non singulière au voisinage de la solution. L'avantage de cette transformation est que la nouvelle équation a la même solution que l'équation d'origine en évitant le problème de singularité.

Notations and Definitions

CHAPTER 1

Notations and Definitions

1.1 Glossary of Symbols

Vectors

$\mathbb{R}^n$	real n-dimensional space
$\mathbb{R}^{n\times n}$	the real vector space of $n \times n$ matrices with real entries
$x = (x^{(1)}, \ldots, x^{(n)})^T$	a column vector with components $x^{(i)}, i = 1, \ldots, n$
x^T	the transpose of x
$\{x^{(k)}\}_{k\in\mathbb{N}}$	a sequence of vectors of $\mathbb{C}^n$
$(x, y) = \bar{y}^T x = \sum_{i=1}^{n} \bar{y}^{(i)} x^{(i)}$	the euclidean scalar product
$\|\|x\|\|_2 = (x, x)^{\frac{1}{2}}$	the euclidean norm
$\|\|.\|\|_p$	the l_p-norm on $\mathbb{R}^n$, $1 \le p \le \infty$
e	a column vector with components 1, $e = [1\ 1\ \ldots\ 1]^T$

Matrices

$A = (a_{ij})$	an $n \times n$ matrix with elements a_{ij}
A^{-1}	the inverse of A
A^T	the transpose of A
A^H	the conjugate transpose of A, obtained from A by taking the transpose and then taking the complex conjugate of each entry (i.e., negating their imaginary parts but not their real parts).
$\det(A)$	the determinant of A
A^k	the k^{th} power of A
A^+	Moore-Penrose pseudoinverse of A
$\{A_k\}$	a sequence of matrices
$\sigma(A)$	the spectrum of A (set of all eigenvalues λ_i of A)
$\rho(A)$	the spectral radius of A, $\rho(A) = \max_{\lambda \in \sigma(A)} (\lvert\lambda\rvert)$
$\lvert\lvert A\rvert\rvert$	an arbitrary norm of A
$\lvert\lvert A\rvert\rvert_p$	the l_p-norm of A, $1 \leq p \leq \infty$
$\text{rank}(A)$	the rank of A
I_n	the nxn identity matrix
$A \leq B$	the partial ordering $a_{ij} \leq b_{ij},\ i, j = 1, \ldots, n$
$\text{diag}(a_1, \ldots, a_n)$	a diagonal matrix with elements $a_1, \ldots, a_n$ on its diagonal
$A\ o\ B$	the Hadamard product of two matrices A and B of the same size, $A\ o\ B = [a_{ij}.b_{ij}]$

Let $A \in \mathbb{R}^{n \times n}$. Then A is

- symmetric if A^T=A.
- diagonal if $a_{i,j} = 0$ for $i \neq j$.
- lower triangular if $a_{i,j} = 0$ for $i < j$.
- upper triangular if $a_{i,j} = 0$ for $i > j$.
- orthogonal if $AA^T = A^T A = I_N$.
- nonnegative if $A = (a_{ij}) \geq 0$.

- positive if $A = (a_{ij}) > 0$.
- a Z-matrix if all of its off-diagonal elements are non-positive, A can be expressed as $sI - B$ with $B \geq 0$.
- a nonsingular M-matrix if A is a Z-matrix and $s > \rho(B)$, where $\rho(B)$ is the spectral radius of B.

1.2 Notations and Definitions

In this section, we introduce some notations and definitions to be used throughout this thesis.

1.2.1 Vector sequences

Definition: Let $\{s^{(k)}\}_{k\in\mathbb{N}}$ be a sequence of vectors of $\mathbb{R}^n$. Define the finite differences as

$$\Delta s^{(k)} = s^{(k+1)} - s^{(k)}, \qquad k = 0, 1, 2, \cdots, \text{for } i = 1,$$
$$\Delta^i s^{(k)} = \Delta^{i-1} s^{(k+1)} - \Delta^{i-1} s^{(k)}, \qquad k = 0, 1, 2, \cdots, \text{for } i > 1.$$

1.2.2 Convergence speed

We will be dealing with iterative methods in this thesis. The overall cost of the algorithm depends on the number of arithmetic operations required at each step and the number of iterations needed to reach numerical convergence. Then, it is essential to analyze the convergence speed of algorithms and the number of steps needed for convergence. An important difference in the convergence speed is made by linearly and superlinearly convergent algorithms; see [13, 50].

Definition: We say that a vector sequence $\{s^{(k)}\}_{k\in\mathbb{N}} \in \mathbb{R}^n$ *converges* to $s^* \in \mathbb{R}^n$ if

$$\lim_{k\to\infty} \|s^{(k)} - s^*\| = 0.$$

Definition: If there exists a constant $K \in (0, 1)$ and a scalar $k_0 \geq 0$ such that for all $k \geq k_0$,

$$\|s^{(k+1)} - s^*\| \leq K\|s^{(k)} - s^*\|,$$

then the sequence $s^{(k)}$ converges *linearly* to s^*.

Definition: If for a sequence $\{c^{(k)}\}$ which tends to 0,

$$\|s^{(k+1)} - s^*\| \leq c^{(k)} \|s^{(k)} - s^*\|,$$

for all k, then $s^{(k)}$ converges *superlinearly* to s^*.

The next definition is used to distinguish superlinear rates of convergence.
Definition: We say that the sequence $s^{(k)}$ converges with order p to s^* if there exist constants $p > 1$, $K > 0$ and $k_0 \geq 0$ for all $k \geq k_0$ and

$$\|s^{(k+1)} - s^*\| \leq K \|s^{(k)} - s^*\|^p,$$

In particular, convergence with order

- p = 2 is called *quadratic* convergence,
- p = 3 is called *cubic* convergence,
- etc.

Definition: Let $\{s^{(k)}\}_{k\in\mathbb{N}}$ and $\{t^{(k)}\}_{k\in\mathbb{N}}$ be two real convergent sequences to the same limit s^*. We say that $\{t^{(k)}\}_{k\in\mathbb{N}}$ *converges faster* to s^* than $\{s^{(k)}\}_{k\in\mathbb{N}}$ if

$$\lim_{k\to\infty} \frac{\|t^{(k)} - s^*\|}{\|s^{(k)} - s^*\|} = 0.$$

Definition: Let $\{s^{(k)}\}_{k\in\mathbb{N}}$ be a real divergent sequence. If the new sequence $\{t^{(k)}\}_{k\in\mathbb{N}}$ converges to a limit s^*, we call s^* the *anti-limit* of the sequence.

1.2.3 Moore–Penrose pseudoinverse

In linear algebra, a pseudoinverse $A^+ \in \mathbb{R}^{n\times m}$ of a matrix $A \in \mathbb{R}^{m\times n}$ is a generalization of the inverse matrix. The most widely known type of matrix pseudoinverse is the Moore–Penrose pseudoinverse.

A Moore–Penrose pseudoinverse A^+ is the unique matrix that satisfies the following conditions [20, 51]:

1. $AA^+A = A$, (AA^+ need not be the general identity matrix, but it maps all column vectors of A to themselves);
2. $A^+AA^+ = A^+$,
3. $(AA^+)^T = AA^+$,

4. $(A^+A)^T = A^+A$.

The Moore–Penrose pseudoinverse exists and is unique: for any matrix A, there is precisely one matrix A^+, that satisfies the four above conditions.

A matrix satisfying the first two conditions of the definition is known as a generalized inverse. Generalized inverses always exist but are not in general unique. Uniqueness is a consequence of the last two conditions.

1.2.4 The Gram-Schmidt Method

Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and let A have n linearly independent columns $a^1, a^2, \ldots, a^n$. Then, there exist an orthogonal matrix $Q \in \mathbb{R}^{m\times n}$ and an upper triangular matrix $R = (r_{i,j}) \in \mathbb{R}^{n\times n}$ such that $A = QR$. For the proof, see [56].

The Classical Gram-Schmidt method (CGS) allows to have a QR factorization of a matrix A. Note that if A has a full rank and $r_{i,i} > 0$, then this factorization is unique. QR factorizations are very useful for both least squares problems and eigenvalue problems. Note $Q = (q^1, \cdots, q^N)$ and $R = (r_{i,j})$, then the following algorithm presents the classical implementation of the Gram-Schmidt Method.

Algorithm of (CGS) method
$r_{1,1} = \|a^1\|_2$; $q^1 = \frac{a^1}{r_{1,1}}$; For $j = 2, \cdots, n$ $r_{i,j} = (a^j, q^i),\ i = 1, \cdots, j-1$; $r_{j,j} = \|a^j - \sum_{i=1}^{j-1} r_{i,j} q^i\|_2$; $q^j = \frac{(a^j - \sum_{i=1}^{j-1} r_{i,j} q^i)}{r_{j,j}}$; end

Table 1.1: Algorithm of the Classical Gram-Schmidt Method (CGS)

Unfortunately, the (CGS) method has very poor numerical properties due to a big loss of orthogonality among the computed q^i. A rearrangement of the calculation, known as modified Gram-Schmidt (MGS), yields a much better computational procedure. In the kth step of (MGS), the kth column of Q and the kth row of R are determined. It is preferable to use the following modified algorithm.

Algorithm of (MGS) method

$r_{1,1} = \|a^1\|_2$;
$q^1 = \frac{a^1}{r_{1,1}}$;
For $j = 2, \cdots, n$
  $w = a^j$;
  For $i = 1, \cdots, j-1$
    $r_{i,j} = (w, q^i)$;
    $w = w - r_{i,j} q^i$;
  end
  $r_{j,j} = \|w\|_2$
  $q^j = \frac{w}{r_{j,j}}$
end

Table 1.2: Algorithm of the Modified Gram-Schmidt Method (MGS)

1.2.5 Operation counts

The unit of measure is "flop", which denotes any of the four elementary scalar operations +, -, $*$, and /. Note that the elapsed time of an algorithm in a particular computing environment may or may not be well predicted by the flop count. Different matrix operations may run at different speeds, depending on the machine used. In particular, matrix multiplications exploits memory storage better that matrix inversion, so it is advisable to make algorithms rich in matrix multiplications rather than inversions. Some matrix computations for real nonsymmetric $n \times n$ matrices are summarized in Tables 1.3 and 1.4. A and B are nonsymmetric, H is symmetric, T is triangular .

Operation	Number of flops
AB	$2n^3$
A^{-1}	$2n^3$
H^{-1}	n^3
T^{-1}	$n^3/3$

Table 1.3: Operation counts for matrix multiplication and inversion.

Factorization/Decomposition	Number of flops
LU factorization with partial pivoting ($PA = LU$)	$2n^3/3$
QR factorization ($A = QR$), $A \in \mathbb{R}^{m\times n}$	$2n^2(m - n/3)$ for R
- for $Q \in \mathbb{R}^{m\times m}$	$4(m^2n - mn^2 + n^3/3)$ for Q
- for $Q \in \mathbb{R}^{m\times n}$	$2n^2(m - n/3)$ for Q
Schur decomposition ($A = QTQ^H$)	$25n^3$ (Q and T), $10n^3$ (T only)

Table 1.4: Operation counts for matrix factorization and decomposition.

Vector Extrapolation Methods

CHAPTER 2

Vector Extrapolation Methods

2.1 Introduction

The aim of this section is to introduce the theory of extrapolation methods, leading to vector extrapolation methods and their main techniques. Passing by two types of extrapolation methods, the scalar type and the vector type. Finally, we arrive at the polynomial-type vector extrapolation techniques, and in particular the Reduced Rank Extrapolation method (RRE) which will be our interest.

An important problem that arises in different areas of science and engineering is that of computing or approximating limits of infinite sequences of vectors $\{x^{(k)}\}$. The elements $x^{(k)}$ of such sequences can appear in the form of partial sums of infinite series, approximations from fixed-point iterations of linear and nonlinear systems of equations, numerical quadrature approximations to finite or infinite range integrals, whether simple or multiple, etc. In most applications, these sequences converge very slowly and this makes their direct use to approximate limits an expensive proposition. Important applications may occur in which these sequences may even diverge. In this case, the direct use of the $x^{(k)}$ to approximate their *antilimits* would be impossible.

Extrapolation methods (or convergence acceleration methods) are techniques which can be applied to such vector sequences that converge to their limits extremely slowly. This is the case, for example, when they result from the finite-difference or finite-element discretizations of continuum problems, where their rates of convergence become worse as the relevant mesh sizes get smaller (hence N becomes larger). In the context of infinite sequences, extrapolation methods are also referred to as *sequence transformations*. These methods transform a sequence of vectors generated by some process to a new one so that it converges faster than the initial sequence. Here, suitable vector extrapolation methods may be applied to accelerate their convergence. An example to these vector sequences is those which are obtained from iterative solution of linear and nonlinear systems of equations. The limits of these sequences are simply the required solutions of these systems. So briefly speaking, an extrapolation method takes a finite (small number) of the $x^{(k)}$ and processes them in some way.

The importance of extrapolation methods as effective computational tools has been widely studied. Two popular representatives, the Richardson extrapolation and the Aitken's Δ^2-process, are discussed in details in most modern textbooks on numerical analysis, and Padè approximants have become an integral part of approximation theory. Finally, international conferences since 1970s partly dedicated to extrapolation methods have been held on a regular basis.

We are interested in vector extrapolation methods and ,in particular, the polynomial methods. A detailed review of vector extrapolation methods, containing the developments up to the early 1980s can be found in the work of Smith, Ford, and Sidi [70]. The most popular vector extrapolation methods can be classified into two categories: the polynomial methods and the ϵ-algorithms. The first category contains the minimal polynomial extrapolation (MPE) method of Cabay and Jackson [10], the reduced rank extrapolation (RRE) method of Eddy [14] and Mesina [48], and the modified minimal polynomial extrapolation (MMPE) method of Sidi et al. *et al*, Brezinski [9] and Pugachev [53]. The second category includes the topological ϵ-algorithm (TEA) method of Brezinski [9], and the scalar and vector ϵ-algorithms (SEA and VEA) of Wynn [78, 77]. Some convergence results and properties of these methods were given in [9, 30, 32, 59, 60, 63, 65, 69, 70]. These methods do not require an explicit knowledge of how the sequence is generated, and consequently can be directly applied for solving linear and nonlinear systems where the Jacobian of the function is not needed for those which are nonlinear.

Two polynomial-type vector extrapolation methods which have been proven [62] to be more timewise efficient and numerically stable convergence accelerators than the epsilon algorithms are the minimal polynomial extrapolation (MPE) and the reduced rank extrapolation (RRE). These convergence accelerators are very efficient in solving large and sparse nonlinear systems of equations that arise in different areas of sciences and engineering. Few difficulties occur in their numerical implementation since their definitions include some linear least-squares problems. The number of equations in these problems is equal to the dimension of the vectors in the given sequence which may be very large leading to a very large matrix of the least-squares problem. Consequently, this requires to store a large rectangular matrix in memory making the MPE and RRE somewhat expensive in both storage and time. The solution of the least-squares problem was recovered in [68] by solving the corresponding normal equations which costs less than using least-squares packages.

Detailed convergence analysis for MPE and RRE have been presented in [63, 65, 67]. Both MPE and RRE, when applied to linearly generated vector sequences, are very related to some well-known Krylov subspace methods [65]. In particular, when applied to linear systems of equations starting with

the same initial approximation, Sidi [65] showed that the MPE and RRE are equivalent to the Arnoldi method [57] and generalized minimal residual method (GMRES) [58], respectively. We will be interested in nonlinear systems of equations in this thesis.

2.2 The theory of extrapolation

Let $\{s^{(k)}\}_{k\in\mathbb{N}}$ be a sequence of real or complex numbers which converges to s. The idea is to transform the sequence $\{s^{(k)}\}$ into another sequence $\{t^{(k)}\}_{k\in\mathbb{N}}$ and denote by T such a transformation. In case the original sequence is divergent, the sequence transformation acts as an extrapolation method to the anti-limit s^*.

For example,

$$t^{(k)} = \frac{s^{(k)} + s^{(k+1)}}{2}, \qquad k = 0, 1, \ldots \tag{2.1}$$

or

$$t^{(k)} = \frac{s^{(k)} s^{(k+2)} - (s^{(k+1)})^2}{s^{(k+2)} - 2s^{(k+1)} + s^{(k)}}, \qquad k = 0, 1, \ldots \tag{2.2}$$

(which is the well-known Aitken's Δ^2-process in [1] and which will be discussed later).

The new sequence $\{t^{(k)}\}$ must exhibit, at least for some particular classes of convergent sequences $\{s^{(k)}\}$, the following properties:

1. $\{t^{(k)}\}$ must converge.
2. $\{t^{(k)}\}$ must converge to the same limit as $\{s^{(k)}\}$,which means that T is regular for the sequence $\{s^{(k)}\}$.
3. $\{t^{(k)}\}$ must converge to s faster than $\{s^{(k)}\}$, that is $\lim_{k\to\infty} \frac{t^{(k)}-s^*}{s^{(k)}-s^*} = 0$.

- In case (2), we say that the transformation T is *regular* for the sequence $\{s^{(k)}\}$.
- In case (3), we say that the transformation T *accelerates the convergence* of the sequence $\{s^{(k)}\}$, i.e. the sequence $\{t^{(k)}\}$ *converges faster than* $\{s^{(k)}\}$.

Note that these properties do not hold for all converging sequences $\{s^{(k)}\}$ and in particular, the last one since, as proved by Delahaye and Germain-Bonne [12] , a universal transformation T accelerating all the converging sequences

cannot exist. This negative result also holds for some classes of sequences such as the set of monotone sequences or that of logarithmic sequences (that is such that $\lim_{k\to\infty}(s^{(k+1)}-s^*)/(s^{(k)}-s^*)=1$). Thus, this negative result means that it will be always interesting to find and to study new sequence transformations since, in fact, each of them is only able to accelerate the convergence of certain classes of sequences.

Looking back to the first example (2.1), this is a linear transformation for which, for all converging sequence $\{s^{(k)}\}$, the sequence $\{t^{(k)}\}$ converges and has the same limit as $\{s^{(k)}\}$. These kinds of linear transformations, called summation processes,have been widely studied and the transformations named after Euler, Cesaro, Hausdorff, Abel, and others, are well known. Now, to find the class of sequences which the example (2.1) accelerates, we have

$$\frac{t^{(k)}-s^*}{s^{(k)}-s^*}=\frac{1}{2}(1+\frac{s^{(k+1)}-s^*}{s^{(k)}-s^*})$$

and thus

$$\lim_{k\to\infty}\frac{t^{(k)}-s^*}{s^{(k)}-s^*}=0$$

if and only if

$$\lim_{k\to\infty}\frac{s^{(k+1)}-s^*}{s^{(k)}-s^*}=-1$$

which shows that this transformation is only able to accelerate the convergence of a very restricted class of sequences. This is mainly the case for all summation processes.

The second sequence transformation in example (2.2) refers to Aitken's Δ^2 process. It can be easily proved that it accelerates the convergence of all the sequences for which it exists $\lambda\in[-1,+1[$ such that

$$\lim_{k\to\infty}\frac{s^{(k+1)}-s^*}{s^{(k)}-s^*}=\lambda$$

which is a much wider class than the sequences accelerated by the first linear transformation. Examples of convergent sequences $\{s^{(k)}\}$ for which the sequence $\{t^{(k)}\}$ obtained by Aitken's process has two accumulation points, are known. But it can also be proved that if such a $\{t^{(k)}\}$ converges, then its limit is the same as the limit of the sequence $\{s^{(k)}\}$, see Tucker [71].

In conclusion, nonlinear sequence transformations usually have better acceleration properties than linear summation processes (that is, they accelerate wider classes of sequences). But, on the other hand, they do not always transform a convergent sequence into another converging sequence and, even if so, both limits can be different.

In this thesis, we shall be mostly interested by nonlinear sequence transformations. Surveys on linear summation processes were given by Joyce [36], Powell and Shah [52] and Wimp [74]. One can also consult Wynn [79], Wimp [75, 76], Niethammer [49], Gabutti [16], Gabutti and Lyness [17] and Walz [72] among others where interesting developments and applications of linear sequence transformations can be found.

There is another problem to be mentioned when using Aitken's process,which is that the computation of $t^{(k)}$ uses $s^{(k)}$, $s^{(k+1)}$ and $s^{(k+2)}$. For some sequences it is possible that $\lim_{k\to\infty} \frac{t^{(k)}-s^*}{s^{(k)}-s^*} = 0$ and that $\lim_{k\to\infty} \frac{t^{(k)}-s^*}{s^{(k+1)}-s^*}$ or $\lim_{k\to\infty} \frac{t^{(k)}-s^*}{s^{(k+2)}-s^*}$ be different from zero. In particular if $\lim_{k\to\infty} \frac{s^{(k+1)}-s^*}{s^{(k)}-s^*} = 0$ then $\{t^{(k)}\}$ obtained by Aitken's process converges faster than $\{s^{(k)}\}$ and $\{s^{(k+1)}\}$ but not always faster than $\{s^{(k+2)}\}$. Thus, in the study of a sequence transformation, it would be better to look at the ratio $(t^{(k)} - s^*)/(s^{(k+q)} - s^*)$ where $s^{(k+q)}$ is the term with the greatest index used in the computation of $t^{(k)}$. However it must he remarked that

$$\frac{t^{(k)} - s^*}{s^{(k+q)} - s^*} = \frac{t^{(k)} - s^*}{s^{(k)} - s^*} \cdot \frac{s^{(k)} - s^*}{s^{(k+1)} - s^*} \cdot \cdots \cdot \frac{s^{(k+q-1)} - s^*}{s^{(k+q)} - s^*}$$

which shows that if $\frac{t^{(k)}-s^*}{s^{(k)}-s^*}$ tends to zero and if $\frac{s^{(k+1)}-s^*}{s^{(k)}-s^*}$ is always away from zero and do not tend to it, then the ratio $\frac{t^{(k)}-s^*}{s^{(k+q)}-s^*}$ also tends to zero. In practice, avoiding a null limit for $\frac{s^{(k+1)}-s^*}{s^{(k)}-s^*}$ is not a severe restriction since, in such a case, $\{s^{(k)}\}$ converges fast enough and does not need to he accelerated.

Now, some interesting properties of sequence transformations on the two preceding examples will be studied. In the study of a sequence transformation the first question to he asked and solved (before those of convergence and acceleration) is an algebraic one: it concerns the so-called kernel of the transformation that is the set of sequences for which $\exists s^*$ such that $\forall k$, $t^{(k)} = s^*$ (in the sequel $\forall k$ would eventually mean $\forall k > N$).

For our linear summation process it is easy to check that its kernel is the set of sequences of the form

$$s^{(k)} = s^* + a(-1)^k \tag{2.3}$$

where a is a scalar.

For Aitken's process the kernel is the set of sequences of the form

$$s^{(k)} = s^* + a\lambda^k \tag{2.4}$$

where a and λ are scalars with $a \neq 0$ and $\lambda \neq 1$.

Thus, obviously, the kernel of Aitken's process contains the kernel of the first linear summation process. In both cases, the kernel depends on some arbitrary parameters, s^* and a in the first case, s^*, a and $\lambda(\neq 1)$ in the second.

If the sequence $\{s^{(k)}\}$ to be accelerated belongs to the kernel of the transformation used then, by construction, we shall have $\forall k,\ t^{(k)} = s^*$.

Of course, usually, s^* is the limit of the sequence $\{s^{(k)}\}$ but this is not always the case and the question needs to be studied. For example, in Aitken's process, s^* is the limit of $\{s^{(k)}\}$ if $|\lambda| < 1$. If $|\lambda| > 1$, $\{s^{(k)}\}$ diverges and s^* is often called its anti-limit. If $|\lambda| = 1$, $\{s^{(k)}\}$ has no limit at all or it only takes a finite number of distinct values and s^* is, in this case, their arithmetical mean.

The two above expressions give the explicit form of the sequences belonging to the respective kernels of our transformations. For that reason we shall call them the explicit forms of the kernel.

However the kernel can also be given in an implicit form that is by means of a relation which holds among consecutive terms of the sequence. Thus, for the first transformation, it is equivalent to write that, $\forall k$

$$s^{(k+1)} - s^* = -(s^{(k)} - s^*)$$

while, for Aitken's process, we have $\forall k$

$$s^{(k+1)} - s^* = \lambda(s^{(k)} - s^*)$$

Solving this difference equation, leads to the explicit form of the kernel. Of course, both forms are equivalent and depend on parameters.

2.3 Scalar extrapolation algorithms

In this section, all sequences are considered to be sequences of real numbers.

Given a sequence $\{s^{(k)}\}_{k\in\mathbb{N}}$ with $\lim_{k\to\infty} s^{(k)} = s^*$. An acceleration transformation constructs a second sequence $\{t^{(k)}\}_{k\in\mathbb{N}}$ that converges faster than the original sequence in the following sense

$$\lim_{k\to\infty} \frac{t^{(k)} - s^*}{s^{(k)} - s^*} = 0.$$

Note that if the sequence is divergent, the sequence transformation acts as an extrapolation method to the anti-limit s^*.

2.3.1 The Aitken's Δ^2 -Method

One of the most famous methods for accelerating the convergence of a given sequence. In numerical analysis, Aitken's Δ^2- process is a series acceleration method, used for accelerating the rate of convergence of a sequence. It

is named after Alexander Aitken, who introduced this method in 1926. Its early form was known to Seki Kôwa (end of 17th century) and was found for rectification of the circle, i.e. the calculation of π. It is most useful for accelerating the convergence of a sequence that is converging linearly.

Definition

Let $\{s^{(k)}\}_{k\in\mathbb{N}}$ be a sequence of real or complex numbers which converges to s of the following form

$$s^{(k)} = s^* + a_1(\lambda_1)^k + a_2(\lambda_2)^k,\ k = 0, 1, \cdots \tag{2.5}$$

where $0 < |\lambda_2| < |\lambda_1| < 1$ and $a_1 a_2 \neq 0$. Then

$$\frac{s^{(k+1)} - s^*}{s^{(k)} - s^*} = \frac{a_1(\lambda_1)^{k+1} + a_2(\lambda_2)^{k+1}}{a_1(\lambda_1)^k + a_2(\lambda_2)^k}. \tag{2.6}$$

We have

$$\frac{s^{(k+1)} - s^*}{s^{(k)} - s^*} = \lambda_1 + O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^k\right).$$

Using equation (2.6), we have

$$\frac{s^{(k+2)} - s^{(k+1)}}{s^{(k+1)} - s^{(k)}} = \lambda_1 + O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^k\right). \tag{2.7}$$

Now, the Aitken's Δ^2- process transforms the sequence $\{s^{(k)}\}$ into another sequence $\{t^{(k)}\}_{k\in\mathbb{N}}$ defined by

$$t^{(k)} = \frac{s^{(k)}s^{(k+2)} - (s^{(k+1)})^2}{s^{(k+2)} - 2s^{(k+1)} + s^{(k)}} = s^{(k)} - \frac{\Delta s^{(k)}}{\Delta^2 s^{(k)}}, \qquad k = 0, 1, \ldots \tag{2.8}$$

where
$\Delta s^{(k)} = s^{(k+1)} - s^{(k)}$, and $\Delta^2 s^{(k)} = \Delta s^{(k+1)} - \Delta s^{(k)} = s^{(k+2)} - 2s^{(k+1)} + s^{(k)}$.

The first formula above is numerically unstable since, when the terms are close to s^* , cancellation arises in the numerator and in the denominator. Of course, such a cancellation also occurs in the second formula, however only in the computation of a correcting term to $s^{(k)}$. Thus, cancellation appears as a second-order error and it follows that the second formula is more stable than the first one, which is only used for theoretical purposes.

Convergence of the sequence $\{t^{(k)}\}_{k\in\mathbb{N}}$ and convergence acceleration properties of the Aitken process are already discussed in Section 2.2.

The new sequence $t^{(k)}$ satisfies

$$\lim_{k\to\infty}\frac{t^{(k)}-s^*}{s^{(k)}-s^*}=0.$$

On the other hand, Aitken's Δ^2- process can be expressed as

$$t^{(k)}=s^{(k)}-\frac{\Delta s^{(k)}}{\Delta^2 s^{(k)}}=s^{(k)}-\Delta s^{(k)}(\Delta^2 s^{(k)})^{-1}\Delta s^{(k)}$$

This shows that $t^{(k)}$ can be written as a ratio of two determinants

$$t^{(k)}=\frac{\begin{vmatrix} s^{(k)} & s^{(k+1)} \\ \Delta s^{(k)} & \Delta s^{(k+1)} \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ \Delta s^{(k)} & \Delta s^{(k+1)} \end{vmatrix}}. \tag{2.9}$$

Interpretation of Aitken's Δ^2-process

Also, $t^{(k)}$ can be expressed as a linear combination

$$t^{(k)}=\eta_0^{(k)}s^{(k)}+\eta_1^{(k)}s^{(k+1)}, \tag{2.10}$$

where $\eta_0^{(k)}=\frac{\Delta s^{(k+1)}}{\Delta^2 s^{(k)}}$ and $\eta_1^{(k)}=-\frac{\Delta s^{(k)}}{\Delta^2 s^{(k)}}$. Consequently, $\eta_0^{(k)}$ and $\eta_1^{(k)}$ satisfy the following linear system of equations

$$\begin{cases} \eta_0^{(k)} & + & \eta_1^{(k)} & = 1 \\ \eta_0^{(k)}\Delta s^{(k)} & + & \eta_1^{(k)}\Delta s^{(k+1)} & = 0. \end{cases} \tag{2.11}$$

2.3.2 Transformation of Shanks

The transformation of Shanks [61] is a generalisation of the Aitken's Δ^2-process using $m+1$ terms of the sequence $\{s^{(k)}\}_{k\in\mathbb{N}}$. The generalisation of equation (2.9) is given by

$$t^{(k,m)}=\frac{\begin{vmatrix} s^{(k)} & s^{(k+1)} & \dots & s^{(k+m)} \\ \Delta s^{(k)} & \Delta s^{(k+1)} & \dots & \Delta s^{(k+m)} \\ \vdots & \vdots & & \vdots \\ \Delta s^{(k+m-1)} & \Delta s^{(k+m)} & \dots & \Delta s^{(k+2m-1)} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \dots & 1 \\ \Delta s^{(k)} & \Delta s^{(k+1)} & \dots & \Delta s^{(k+m)} \\ \vdots & \vdots & & \vdots \\ \Delta s^{(k+m-1)} & \Delta s^{(k+m)} & \dots & \Delta s^{(k+2m-1)} \end{vmatrix}}. \tag{2.12}$$

Using Cramer's formula, a definition of $t^{(k,m)}$ can be deduced with the following form

$$t^{(k,m)} = \sum_{i=0}^{m} \eta_i^{(k)} s^{(k+i)} \tag{2.13}$$

where η_j can be obtained by the following linear system of equations

$$\begin{cases} \eta_0^{(k)} & + & \eta_1^{(k)} & + & \cdots & + & \eta_m^{(k)} & = 1 \\ \eta_0^{(k)}\Delta s^{(k)} & + & \eta_1^{(k)}\Delta s^{(k+1)} & + & \cdots & + & \eta_m^{(k)}\Delta s^{(k+m)} & = 0 \\ \vdots & & \vdots & & \cdots & & \vdots & \\ \eta_0^{(k)}\Delta s^{(k+m-1)} & + & \eta_1^{(k)}\Delta s^{(k+m)} & + & \cdots & + & \eta_m^{(k)}\Delta s^{(k+2m-1)} & = 0. \end{cases} \tag{2.14}$$

2.4 Vector extrapolation techniques

Vector extrapolation methods are convergence acceleration methods which are usually obtained by an extrapolation procedure.

2.4.1 Notation and description of algorithms

Let B be a normed linear space defined over the field of complex numbers, and denote the norm associated with B by $||.||$. In case B is also an inner product space, we adopt the following convention for the homogeneity property of the inner product : For $y, z \in B$ and α, β being complex numbers, the inner product $(.,.)$ is defined such that $(\alpha y, \beta z) = \bar{\alpha}\beta(y, z)$. The norm in this case is the one induced by the inner product, i.e., for a vector $x \in B$, $||x|| = \sqrt{(x, x)}$.

Let us consider a sequence of vectors $s^{(i)}$, $i = 0, 1, \ldots$, in B. We shall assume that

$$s^{(k)} \sim s^* + \sum_{i=1}^{\infty} v_i \lambda_i^k \quad \text{as} \quad k \to \infty, \tag{2.15}$$

where s^* and v_i are vectors in B, and λ_i, $i = 1, 2, \ldots$, are scalars, such that

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \tag{2.16}$$

and satisfying $\lambda_i \neq 1$, $i = 1, 2, \ldots$, and $\lambda_i \neq \lambda_j$ if $i \neq j$. In addition, we assume that there can be only a finite number of λ_i whose moduli are equal. Without loss of generality, we assume in (2.15) that $v_i \neq 0$, $\lambda_i \neq 0$ for all $i \geq 1$. The meaning of (2.15) is that for any integer $N > 0$, there exist a positive constant K and a positive integer k_o that depend only on N, such that for every $k \geq k_O$, the vector

$$\tilde{v}_N(k) = (s^{(k)} - s^* - \sum_{i=1}^{N-1} v_i \lambda_i^k)/\lambda_N^k, \tag{2.17}$$

satisfies

$$||\tilde{v}_N(k)|| \leq K. \tag{2.18}$$

If $|\lambda_1| < 1$, then $\lim_{k\to\infty} s^{(k)}$ exists and is simply s^*. If $|\lambda_1| \geq 1$, then $\lim_{k\to\infty} s^{(k)}$ does not exist, and s^* is said to be the anti-limit of the sequence $s^{(k)}$, $k = 0, 1, \ldots$ Our aim is to find a good approximation to s^*, whether s^* is the limit or the anti-limit of the sequence, from a relatively small number of the vectors $s^{(i)}$, $i = 0, 1, \ldots$. To this effect several vector extrapolation methods have been proposed. Some of these methods were surveyed and tested numerically in [70, 69], namely the MPE,the MMPE, the RRE, the SEA, the VEA and the TEA. These methods were analyzed for their convergence and stability properties. Their analysis were carried out for sequences of the form (2.15).

2.4.2 The polynomial methods

A brief description of the main polynomial-type vector extrapolation methods (RRE, MPE, and MMPE) using the generalized residual will be introduced. Also their application to solve linear and nonlinear systems of equations will be explained supplied with some theoretical results.

Let k be a positive integer less than or equal to the dimension of the space B and let $\{s^{(k)}\}_{k\in\mathbb{N}}$ be a sequence of vectors in $\mathbb{R}^N$. Define the first and the second forward differences of $s^{(k)}$, Δ and Δ^2 respectively, by

$$\Delta s^{(k)} = s^{(k+1)} - s^{(k)} \quad \text{and} \quad \Delta^2 s^{(k)} = \Delta s^{(k+1)} - \Delta s^{(k)}\ ,\ k = 0, 1, \ldots$$

When applied to the vector sequence $\{s^{(k)}\}$, the extrapolation methods namely the MPE, the RRE, and the MMPE, produce an approximation $t^{(k)}$ of the limit or the antilimit of $\{s^{(k)}\}_{k\in\mathbb{N}}$; see [63]. Clearly, $t^{(k)}$ will be different for each method.

Let this transformation be T_k defined as follows

$$\begin{aligned} T_k\ &: \mathbb{R}^N \longrightarrow \mathbb{R}^N\ , \\ &\quad s^{(k)} \longrightarrow t^{(k,q)} \end{aligned}$$

with

$$t^{(k,q)} = s^{(q)} + \sum_{i=1}^{k} a_i^{(q)} g_i(q), \quad q \geq 0, \tag{2.19}$$

where the coefficients $a_i^{(q)}$ are scalars and $(g_i(q))_q$, the auxiliary vector sequences for these extrapolation methods are given by

$$g_i(q) = \Delta s^{(q+i-1)}, \quad \text{for } i = 1, \ldots, k\ ;\ q \geq 0$$

And denote by $\tilde{T}_k$, the new transformation produced from T_k, by

$$\tilde{t}^{(k,q)} = s^{(q+1)} + \sum_{i=1}^{k} a_i^{(q)} g_i(q+1), \quad q \geq 0, \tag{2.20}$$

Now, we can define the generalized residual of $t^{(k,q)}$ by

$$\tilde{r}(t^{(k,q)}) = \tilde{t}^{(k,q)} - t^{(k,q)} \tag{2.21}$$

$$= \Delta s^{(q)} + \sum_{i=1}^{k} a_i^{(q)} g_i(q). \tag{2.22}$$

Note that the coefficients $a_i^{(q)}$ are obtained from the orthogonality relation

$$\tilde{r}(t^{(k,q)}) \perp \text{span}\{y_1^{(q)}, y_2^{(q)}, \ldots, y_k^{(q)}\}, \tag{2.23}$$

where,

$$y_i^{(q)} = \begin{cases} \Delta s^{(q+i-1)}, & \text{for the MPE} \\ \Delta^2 s^{(q+i-1)}, & \text{for the RRE} \\ y_i, & \text{for the MMPE} \end{cases}$$

where $\{y_1, y_2, \ldots, y_k\}$ are arbitrary linearly independent vectors of $\mathbb{R}^N$.

Denote by $W_{k,q}$ and $Y_{k,q}$ the subspaces defined by

$$W_{k,q} = \text{span}\{\Delta^2 s^{(q)}, \ldots, \Delta^2 s^{(q+k-1)}\} \quad \text{and} \quad Y_{k,q} = \text{span}\{y_1^{(q)}, y_2^{(q)}, \ldots, y_k^{(q)}\}.$$

Then, from (2.22) and (2.23), the generalized residuals satisfies the following conditions

$$\tilde{r}(t^{(k,q)}) - \Delta s^{(q)} \in W_{k,q}$$

and

$$\tilde{r}(t^{(k,q)}) \perp Y_{k,q}.$$

These conditions show that the generalized residual $\tilde{r}(t^{(k,q)})$ is obtained by projecting the vector $\Delta s^{(q)}$ onto the subspace $W_{k,q}$, orthogonally to $Y_{k,q}$. And in a matrix form, $\tilde{r}(t^{(k,q)})$ can be written as

$$\tilde{r}(t^{(k,q)}) = \Delta s^{(q)} - \Delta^2 S_{k,q} \Delta^2 S_{k,q}^{+} \Delta s^{(q)}, \tag{2.24}$$

where $\Delta^2 S_{k,q}^{+}$denotes the Moore-Penrose generalized inverse of $\Delta^2 S_{k,q}$ and it is defined by

$$(Y_{k,q}^T \Delta^2 S_{k,q})^{-1} Y_{k,q}^T, \tag{2.25}$$

with

$$\Delta^i S_{k,q} = [\Delta^i s^{(q)}, \ldots, \Delta^i s^{(q+k-1)}], \quad i = 1,2.$$

Note that $\tilde{r}(t^{(k,q)})$ is well-defined if and only if $\det(Y_{k,q}^T \Delta^2 S_{k,q}) \neq 0$ which requires the matrices $Y_{k,q}$ and $\Delta^2 S_{k,q}$ to be full rank.

Consequently, this leads to the existence and uniqueness of $t^{(k,q)}$ and with these notations and using Schur complements, $t^{(k,q)}$ can be written in a matrix form as

$$\begin{aligned} t^{(k,q)} &= s^{(q)} - \Delta S_{k,q} (Y_{k,q}^T \Delta^2 S_{k,q})^{-1} Y_{k,q}^T \Delta s^{(q)} \\ &= s^{(q)} - \Delta S_{k,q} \Delta^2 S_{k,q}^{+} \Delta s^{(q)}. \end{aligned}$$

It is clear that $t^{(k,q)}$ exists if and only if the $k \times k$ matrix $Y_{k,q}^T \Delta^2 S_{k,q}$ is nonsingular; see [65] for the conditions to be satisfied for this.

Another expression of this approximation can be given by

$$t^{(k,q)} = \sum_{j=0}^{k} \eta_j^{(k)} s^{(q+j)} \tag{2.26}$$

subject to

$$\sum_{j=0}^{k} \eta_j^{(k)} = 1, \tag{2.27}$$

and

$$\sum_{j=0}^{k} \beta_{i,j} \eta_j^{(k)} = 0, \qquad \text{for} \quad i = 0,1,...,k-1, \tag{2.28}$$

where the scalars $\beta_{i,j} \in \mathbb{R}$ are defined by

$$\beta_{i,j} = \begin{cases} (\Delta s^{(q+i)}, \Delta s^{(q+j)}) & \text{for the MPE,} \\ (\Delta^2 s^{(q+i)}, \Delta s^{(q+j)}) & \text{for the RRE,} \\ (y_{i+1}, \Delta s^{(q+j)}) & \text{for the MMPE,} \end{cases} \tag{2.29}$$

for $i = 0,1,...,k-1$ and $j = 0,1,...,k$. Again, $\{y_1, y_2, \ldots, y_k\}$ is a set of linearly independent vectors of $\mathbb{R}^N$ which are often chosen to be the canonical vectors in some order; see e.g., [33].

It follows from (2.26), (2.27), and (2.28) that $t^{(k,q)}$ can also be expressed as a ratio of two determinants as follows

$$t^{(k,q)} = \frac{\begin{vmatrix} s^{(q)} & s^{(q+1)} & \dots & s^{(q+k)} \\ \beta_{0,0} & \beta_{0,1} & \dots & \beta_{0,k} \\ \vdots & \vdots & & \vdots \\ \beta_{k-1,0} & \beta_{k-1,1} & \dots & \beta_{k-1,k} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \dots & 1 \\ \beta_{0,0} & \beta_{0,1} & \dots & \beta_{0,k} \\ \vdots & \vdots & & \vdots \\ \beta_{k-1,0} & \beta_{k-1,1} & \dots & \beta_{k-1,k} \end{vmatrix}} \tag{2.30}$$

Note that the determinant in the numerator of (2.30) is the vector obtained by expanding this determinant with respect to its first row by the classical rule. While the determinant in the denominator is equal to $\det(Y_{k,q}^T \Delta^2 S_{k,q})$ which is assumed to be nonzero. The computation of the approximation $t^{(k,q)}$ needs the values of the terms $s^{(q)}, s^{(q+1)}, s^{(q+k+1)}$, and can be achieved using one of the algorithms proposed in [15] and [34].

Now , again the following matrices $Y_{k,q}$ and $\Delta^i S_{k,q}$ are given as before as

$$Y_{k,q} = [y_1^{(q)}, y_2^{(q)}, \dots, y_k^{(q)}]$$

and

$$\Delta^i S_{k,q} = [\Delta^i s^{(q)}, \dots, \Delta^i s^{(q+k-1)}], \quad i = 1,2.$$

If we replace, in the numerator and the denominator of (2.30) , each column j, for $j = k+1, k+2, \dots$, by the difference with column $j-1$, we get the following expression:

$$t^{(k,q)} = \frac{\begin{vmatrix} s^{(q)} & \Delta S_{k,q} \\ Y_{k,q}^T \Delta s^{(q)} & Y_{k,q}^T \Delta^2 S_{k,q} \end{vmatrix}}{\left| Y_{k,q}^T \Delta^2 S_{k,q} \right|} \tag{2.31}$$

With these notations and using Schur complements, $t^{(k,q)}$ can be written in a matrix form as

$$t^{(k,q)} = s^{(q)} - \Delta S_{k,q} (Y_{k,q}^T \Delta^2 S_{k,q})^{-1} Y_{k,q}^T \Delta s^{(q)}, \tag{2.32}$$

where $t^{(k,q)}$ exists and is unique if and only if $\det(Y_{k,q}^T \Delta^2 S_{k,q}) \neq 0$. For varying values of k and q, the computation of $t^{(k,q)}$ can be done by some of the algorithms proposed by Ford and Sidi in [65].

Implementation

From a perspective implementation, we are interested only in the case when q is kept fixed. Accordingly, from now on, we set $q = 0$ and denote the vector $t^{(0,k)}$ by $t^{(k)}$; $\Delta^i S_{k,0}$ by $\Delta^i S_k$.

The linear system (2.27)-(2.28) can be written as

$$\left\{\begin{array}{lllllllll} \eta_0^{(k)} & + & \eta_1^{(k)} & + & \ldots & + & \eta_k^{(k)} & = & 1 \\ \eta_0^{(k)}(y_0,\Delta s^{(0)}) & + & \eta_1^{(k)}(y_0,\Delta s^{(1)}) & + & \ldots & + & \eta_k^{(k)}(y_0,\Delta s^{(k)}) & = & 0 \\ \eta_0^{(k)}(y_1,\Delta s^{(0)}) & + & \eta_1^{(k)}(y_1,\Delta s^{(1)}) & + & \ldots & + & \eta_k^{(k)}(y_1,\Delta s^{(k)}) & = & 0 \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ \eta_0^{(k)}(y_{k-1},\Delta s^{(0)}) & + & \eta_1^{(k)}(y_{k-1},\Delta s^{(1)}) & + & \ldots & + & \eta_k^{(k)}(y_{k-1},\Delta s^{(k)}) & = & 0. \end{array}\right. \tag{2.33}$$

Introduce the scalars $\gamma_i^{(k)}$, for $i = 0,\ldots,k$, defined by $\gamma_i^{(k)} = \dfrac{\eta_i^{(k)}}{\eta_k^{(k)}}$. In this case, we have

$$\eta_i^{(k)} = \frac{\gamma_i^{(k)}}{\sum_{i=0}^{k}\gamma_i^{(k)}} \text{ for } i = 0,\ldots,k-1 \text{ and } \gamma_k^{(k)} = 1. \tag{2.34}$$

With this notation, the linear system (2.33) becomes

$$\left\{\begin{array}{lllllllll} \gamma_0^{(k)}(y_0,\Delta s^{(0)}) & + & \gamma_1^{(k)}(y_0,\Delta s^{(1)}) & + & \ldots & + & \gamma_{k-1}^{(k)}(y_0,\Delta s^{(k-1)}) & = & -(y_0,\Delta s^{(k)}), \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ \gamma_0^{(k)}(y_{k-1},\Delta s^{(0)}) & + & \gamma_1^{(k)}(y_{k-1},\Delta s^{(1)}) & + & \ldots & + & \gamma_{k-1}^{(k)}(y_{k-1},\Delta s^{(k-1)}) & = & -(y_{k-1},\Delta s^{(k)}). \end{array}\right.$$

This system can be written as the following form

$$(Y_k^T\,\Delta S_k)\,\gamma^{(k)} = -Y_k^T\,\Delta s^{(k)}, \tag{2.35}$$

where $\gamma^{(k)} = (\gamma_0^{(k)},\ldots,\gamma_{k-1}^{(k)})^T$ and $\Delta S_k = (\Delta s^{(0)},\ldots,\Delta s^{(k-1)})$.

Assume now that the coefficients $\eta_0^{(k)},\ldots,\eta_k^{(k)}$ have been calculated and introduce the new variables

$$\begin{aligned} &\xi_0 = 1-\eta_0, \\ &\xi_j = \xi_{j-1} - \eta_j \quad \text{for } j = 1,\ldots,k-1. \\ &\xi_{k-1}^{(k)} = \eta_k^{(k)}. \end{aligned} \tag{2.36}$$

Then, the vector $t^{(k)}$ can be expressed as the following

$$t^{(k)} = s^{(0)} + \sum_{j=0}^{k-1} \xi_j^{(k)} \Delta s^{(j)} = s^{(0)} + \Delta S_k \xi^{(k)}, \tag{2.37}$$

where $\xi = (\xi_0, \ldots, \xi_{q-1})^T$.

We remark that in order to determine the $\eta_i^{(k)}$, we must first calculate the $\gamma_i^{(k)}$ by solving the linear system of equations (2.35). Using (2.21) and (2.37), the generalized residual $\tilde{r}(t^{(k)})$ can be expresses as

$$\tilde{r}(t^{(k)}) = \sum_{i=0}^{k} \eta_i^{(k)} \Delta s^{(i)} = \Delta S_{k+1} \eta^{(k)}$$

2.4.3 The RRE method

Definition

The Reduced Rank Extrapolation (RRE) method was proposed by Eddy [14] and Mesina [48] in the 1970's. They defined the scalars $\beta_{i,j}$ as follows

$$\beta_{i,j} = (\Delta^2 s^{(q+i)}, \Delta s^{(q+j)}). \tag{2.38}$$

and

$$Y_{k,q} = \Delta^2 S_{k,q} \tag{2.39}$$

Then, the transformation of the RRE method can be deduced from (2.32) as

$$t_{RRE}^{(k,q)} = s^{(q)} - \Delta S_{k,q} \Delta^2 S_{k,q}^{+} \Delta s^{(q)}. \tag{2.40}$$

where $\Delta^2 S_{k,q}^{+}$ denotes the Moore-Penrose pseudoinverse of $\Delta^2 S_{k,q}$ defined by

$$\Delta^2 S_{k,q} = (\Delta^2 S_{k,q}^T \Delta^2 S_{k,q})^{-1} \Delta^2 S_{k,q}^T.$$

Implementation

From an implementation perspective, we are interested only in the case when q is kept fixed. Accordingly, from now on, we set $q = 0$ and denote the vector $t^{(k,0)}$ by $t^{(k)}$.

We follow the description given by Sidi in [64]. An important feature of this method is that it proceeds through the solution of least-squares problems by QR factorization.

Introduce the following

$$\Delta S_{k+1} = [\Delta s^{(0)}, \ldots, \Delta s^{(k)}] \quad \text{and} \quad \eta^{(k)} = (\eta_0^{(k)}, \ldots, \eta_k^{(k)})^T. \tag{2.41}$$

In view of (2.26), (2.27), and (2.28), $\eta_j^{(k)}$ can be determined by solving the overdetermined linear system

$$\Delta S_{k+1}\eta^{(k)} = 0, \tag{2.42}$$

by the least-squares method subject to the constraint $\sum_{j=0}^{k}\eta_j^{(k)} = 1$. This leads to minimizing the positive definite quadratic form

$$\eta^{(k)\,T}(\Delta S_{k+1})^T\Delta S_{k+1}\eta^{(k)},$$

subject to the same constraint.

It was proved in [64] that $\eta_i^{(k)}$ $(i = 0,\ldots,k)$ can be obtained by solving the following linear system of $(k+2)$ equations

$$\Delta S_{k+1}^T\Delta S_{k+1}\eta^{(k)} = \lambda e, \qquad \sum_{j=0}^{k}\eta_j^{(k)} = 1 \tag{2.43}$$

where $e = (1,\cdots,1)^T \in \mathbb{R}^{k+1}$ and λ is a strictly positive scalar such that

$$\lambda = \eta^{(k)T}\Delta S_{k+1}^T\Delta S_{k+1}\eta^{(k)}. \tag{2.44}$$

Set $d^{(k)} = [d_0^{(k)},\ldots,d_k^{(k)}]^T$, $\lambda = (\sum_{i=0}^{k} d_i^{(k)})^{-1}$, and $\eta^{(k)} = \lambda d^{(k)}$
$(\eta_i^{(k)} = \lambda d_i^{(k)})$. Then $\eta^{(k)}$ can be computed by solving the linear system of equations

$$(\Delta S_{k+1})^T\Delta S_{k+1}d^{(k)} = e. \tag{2.45}$$

Assume that ΔS_{k+1} has full rank. Then it has a QR factorization $\Delta S_{k+1} = Q_kR_k$. This leads to another form of the linear system (2.45)

$$R_k^TR_kd^{(k)} = e.$$

Finally, the approximation $t^{(k)}$ can be expressed as

$$t_{RRE}^{(k)} = s^{(0)} + Q_{k-1}(R_{k-1}\xi^{(k)}),$$

where $\xi^{(k)} = [\xi_0^{(k)},\xi_1^{(k)},\ldots,\xi_{k-1}^{(k)}]^T$ and $\xi_0^{(k)} = 1-\eta_0^{(k)}$; $\quad \xi_j^{(k)} = \xi_{j-1}^{(k)} - \eta_j^{(k)}$ for $j = 1,\ldots,k-1$.

Another expression of $t^{(k)}$ is given by

$$t_{RRE}^{(k)} = s^{(0)} + \sum_{j=0}^{k-1}\xi_j^{(k)}\Delta s^{(j)} = s^{(0)} + \Delta S_k\xi^{(k)}. \tag{2.46}$$

Then, using (2.21) and (2.46), the generalized residual $\tilde{r}(t^{(k)})$ can be written as

$$\tilde{r}(t_{RRE}^{(k)}) = \sum_{i=0}^{k} \eta_i^{(k)} \Delta s^{(i)} = \Delta S_{k+1} \eta^{(k)}.$$

Note that the QR factorization of ΔS_{k+1} is formed by appending one additional column to Q_{k-1} to obtain Q_k, and a corresponding column to R_{k-1} to obtain R_k. This QR factorization can be computed inexpensively by applying the modified Gram-Schmidt process (MGS) to the vectors $s^{(0)}, s^{(1)}, \ldots, s^{(k+1)}$; see [64].

Let Q_k and R_k be as follows

$$Q_k = [q_0|q_1|\ldots|q_k] \quad \text{and} \quad R_k = \begin{bmatrix} r_{00} & r_{01} & r_{02} & \ldots & r_{0k} \\ & r_{11} & r_{12} & \ldots & r_{1k} \\ & & r_{22} & \ldots & r_{2k} \\ & & & \ddots & \vdots \\ & & & & r_{kk} \end{bmatrix}$$

and denote by $u_i = \Delta s^{(i)}$. Then, the modified Gram-Schmidt process can be summarized in the following table:

Algorithm of (MGS) method

$r_{00} = \|u_0\|_2$;
$q_0 = u_0 / r_{00}$;
For $k = 1, 2, \cdots,$
  Set $u_k^{(0)} = u_k$;
  For $j = 0, \cdots, k-1$
    $r_{jk} = (q_j, u_k^{(j)})$;
    $u_k^{(j+1)} = u_k^{(j)} - r_{jk} q_j$.
  End
  $r_{kk} = \|u_k^{(k)}\|_2$
  $q_k = u_k^{(k)} / r_{kk}$.
end

Table 2.1: Algorithm of the Modified Gram-Schmidt Method (MGS)

An algorithm for the RRE method is presented in Table 2.2. Fast, stable, and low memory algorithms can be found in [30, 64, 69].

Algorithm of the RRE method

1. Input: Vectors $s^{(0)}, s^{(1)}, \ldots, s^{(k+1)}$.
2. Compute $\Delta s^{(i)} = s^{(i+1)} - s^{(i)}$, for $i = 0, 1, \ldots, k$.
   Set $\Delta S_{k+1} = [\Delta s^{(0)}, \Delta s^{(1)}, \ldots, \Delta s^{(k)}]$.
   Compute the QR factorization of ΔS_{k+1}, namely,
   $\Delta S_{k+1} = Q_k R_k$.
3. Solve the linear system
   $R_k^T R_k d^{(k)} = e$; where $d^{(k)} = [d_0^{(k)}, d_1^{(k)}, \ldots, d_k^{(k)}]^T$ and $e = [1, 1, \ldots, 1]^T$.
   (This amounts to solving two upper and lower triangular systems.)
   Set $\lambda = (\sum_{i=0}^{k} d_i^{(k)})^{-1}$, $\lambda \in \mathbb{R}^+$.
   Set $\eta_i^{(k)} = \lambda d_i^{(k)}$, for $i = 0, 1, \ldots, k$.
4. Compute $\xi^{(k)} = [\xi_0^{(k)}, \xi_1^{(k)}, \ldots, \xi_{k-1}^{(k)}]^T$ where $\xi_0^{(k)} = 1 - \eta_0^{(k)}$ and $\xi_j^{(k)} = \xi_{j-1}^{(k)} - \eta_j^{(k)}$, $1 \le j \le k-1$.
   Compute $t_{RRE}^{(k)}$ by : $t_{RRE}^{(k)} = s^{(0)} + Q_{k-1}(R_{k-1}\xi^{(k)})$.

Table 2.2: Algorithm of the RRE method

2.4.4 The MPE method

Definition

The Minimal Polynomial Extrapolation (MPE) was proposed by Cabay and Jackson in [10]. They defined the scalars $\beta_{i,j}$ as follows

$$\beta_{i,j} = (\Delta s^{(q+i)}, \Delta s^{(q+j)}) \tag{2.47}$$

and

$$Y_{(k,q)} = \Delta S_{k,q}. \tag{2.48}$$

Then, the transformation of the MPE method can be deduced from (2.32) as

$$t_{MPE}^{(k,q)} = s^{(q)} - \Delta S_{k,q}(\Delta S_{k,q}^T \Delta^2 S_{k,q})^{-1} \Delta S_{k,q}^T \Delta s^{(q)}, \tag{2.49}$$

Implementation

From an implementation perspective, we are interested only in the case when q is kept fixed. Accordingly, from now on, we set $q = 0$ and denote the vector $t^{(k,0)}$ by $t^{(k)}$.

Suppose that ΔS_{k+1} has a full rank, i.e. $\text{rank}(\Delta S_{k+1}) = k+1$. Then, there exist a QR factorization of the matrix ΔS_{k+1}. Let $\Delta S_{k+1} = Q_{k+1}R_{k+1}$, where $Q_{k+1} = [q_0, q_1, \cdots, q_k] \in \mathbb{R}^{N\times(k+1)}$ is an orthogonal matrix and $R_{k+1} \in \mathbb{R}^{(k+1)\times(k+1)}$ is an upper triangular matrix with positive diagonal entries. The matrix Q_{k+1} is obtained from the matrix $Q_k \in \mathbb{R}^{N\times k}$ upon adding the vector column q_k. Similarly, R_{k+1} is obtained from the matrix $R_k \in \mathbb{R}^{k\times k}$ upon adding a row and a column to R_k. This QR factorization can be computed inexpensively by applying the modified Gram-Schmidt process (MGS) to the vectors $s^{(0)}, s^{(1)}, \ldots, s^{(k+1)}$; see Table 2.1.

This factorization can be written as

$$\left(\Delta S_k, \Delta s^{(k)}\right) = \left(Q_k, q_k\right)\begin{pmatrix} R_k & r_k \\ 0 & \rho_k \end{pmatrix}, \tag{2.50}$$

where $r_k \in \mathbb{R}^k$ is formed by the last column of the matrix R_{k+1} without the last element. ρ_k is a scalar corresponding to this element. Developing the right hand side of (2.50), we get

$$\left(\Delta S_k, \Delta s^{(k)}\right) = \left(Q_k R_k, Q_k r_k + \rho_k q_k\right). \tag{2.51}$$

Consider the last column in each side, then

$$\Delta s^{(k)} = Q_k r_k + \rho_k q_k. \tag{2.52}$$

Since $\Delta S_k = Q_k R_k$, multiply each side of (2.52) with ΔS_k^T, then

$$\Delta S_k^T \Delta s^{(k)} = R_k^T Q_k^T \left(Q_k r_k + \rho_k q_k\right). \tag{2.53}$$

Since Q_{k+1} is orthogonal, we get

$$\Delta S_k^T \Delta s^{(k)} = R_k^T r_k. \tag{2.54}$$

The linear system (2.35) can be simplified as

$$\begin{aligned}
& \left(\Delta S_k^T \Delta S_k\right)\gamma^{(k)} = -\Delta S_k^T \Delta s^{(k)} \\
\iff \quad & R_k^T Q_k^T Q_k R_k \gamma^{(k)} = -R_k^T r_k \\
\iff \quad & R_k^T R_k \gamma^{(k)} = -R_k^T r_k \qquad \text{(since } Q \text{ is orthogonal)} \\
\iff \quad & R_k \gamma^{(k)} = -r_k \qquad \text{(since } R_k \text{ is nonsingular)}
\end{aligned}$$

After calculating $\gamma^{(k)}$, we compute $\eta_i^{(q)}$ $(i = 0, \cdots, k)$ from (2.34) and ξ_i $(i = 0, \cdots, k-1)$ from (2.36). In the end, we calculate the approximation $t_{MPE}^{(k)}$ as follows

$$t_{MPE}^{(k)} = s^{(0)} + Q_k R_k \xi^{(k)}. \tag{2.55}$$

Algorithm

Using the above implementation, the algorithm of the MPE method is given in Table 2.3.

Algorithm of the MPE method

1. Input: Vectors $s^{(0)}, s^{(1)}, \ldots, s^{(k+1)}$.
2. Compute $\Delta s^{(i)} = s^{(i+1)} - s^{(i)}$, for $i = 0, 1, \ldots, k$.
   Set $\Delta S_j = [\Delta s^{(0)}, \Delta s^{(1)}, \ldots, \Delta s^{(j)}]$, $j = 0, 1, \ldots$;
   Compute the QR factorization of ΔS_{k+1}, namely,
   $\Delta S_{k+1} = Q_{k+1} R_{k+1}$.
   ($\Delta S_k = Q_k R_k$ is contained in $\Delta S_{k+1} = Q_{k+1} R_{k+1}$)
3. Solve the upper triangular linear system
   $R_k d^{(k)} = -r_k$; where
   $r_k = [r_{0k}, r_{1k}, \ldots, r_{(k-1)k}]^T$ and $d^{(k)} = [d_0^{(k)}, d_1^{(k)}, \ldots, d_{k-1}^{(k)}]^T$;
   Set $d_k^{(k)} = 1$ and calculate $\lambda = (\sum_{i=0}^{k} d_i^{(k)})^{-1}$, $\lambda \in \mathbb{R}^+$.
   Set $\eta_i^{(k)} = \lambda d_i^{(k)}$, for $i = 0, 1, \ldots, k$.
4. Compute $\xi^{(k)} = [\xi_0^{(k)}, \xi_1^{(k)}, \ldots, \xi_{k-1}^{(k)}]^T$ where $\xi_0^{(k)} = 1 - \eta_0^{(k)}$ and $\xi_j^{(k)} = \xi_{j-1}^{(k)} - \eta_j^{(k)}$,
   $1 \le j \le k-1$.
   Compute $t_{MPE}^{(k)}$ by: $t_{MPE}^{(k)} = s^{(0)} + Q_k(R_k \xi^{(k)})$.

Table 2.3: Algorithm of the MPE method

2.4.5 The MMPE method

Definition

The Modified Minimal Polynomial (MMPE) method is defined by Brezinski [9], Pugatchev [53] et Sidi, Ford et Smith [69].

Choose Y_k such that the columns $\{y_0, \cdots, y_{q-1}\}$ form a set of linearly independent vectors of $\mathbb{R}^N$. It is not preferable to use QR decomposition of the matrix Y_k, instead, PLU decomposition will be used.

Implementation

From an implementation perspective, we are interested only in the case when q is kept fixed. Accordingly, from now on, we set $q = 0$ and denote the vector $t^{(k,0)}$ by $t^{(k)}$.

We follow the description given by Sadok and Jbilou in [33]. This new implementation, which is based on an LU factorization with a pivoting strategy, is inexpensive both in time and storage as compared with other extrapolation methods. Suppose that ΔS_{q+1} has full rank, i.e. $\text{rank}(\Delta S_{k+1}) = k+1$. Then, there exist a permutation matrix $P \in \mathbb{R}^{N\times N}$,a unit lower trapezoidal matrix $L_{k+1} \in \mathbb{R}^{N\times(k+1)}$ and an invertible upper triangular matrix $R_{k+1} \in \mathbb{R}^{(k+1)\times(k+1)}$ such that

$$P\Delta S_{k+1} = L_{k+1}R_{k+1}. \tag{2.56}$$

The entries in $P^T L_{k+1}$ will be less than or equal to 1 in magnitude. Let P_{k+1} be the matrix formed by the first $(k+1)$ rows of P, and L'_{k+1} be the unit lower matrix formed by the first $(k+1)$ rows of L_{k+1}. Then, using equation (2.56), we get the following factorization

$$P_{k+1}\Delta S_{k+1} = L'_{k+1}R_{k+1}, \tag{2.57}$$

which can be also written as

$$P_{k+1}\left(\Delta S_k, \Delta s^{(k)}\right) = \begin{pmatrix} L'_k & 0 \\ v_k^T & 1 \end{pmatrix}\begin{pmatrix} R_k & r_k \\ 0 & \rho_k \end{pmatrix}, \tag{2.58}$$

where r_k and v_k are two vector of $\mathbb{R}^k$, and ρ_k is a scalar. r_k corresponds to the first k elements of the last column of R_{k+1}. Developing equation (2.58) leads to

$$\left(P_{k+1}\Delta S_k, P_{k+1}\Delta s^{(k)}\right) = \begin{pmatrix} L'_k R_k & L'_k r_k \\ v_k^T R_k & v_k^T r_k + \rho_k \end{pmatrix}. \tag{2.59}$$

Upon comparing the last columns of the two matrices of (2.59), it follows that

$$\left(P_{k+1}\Delta s^{(k)}\right) = \begin{pmatrix} L'_k r_k \\ v_k^T r_k + \rho_k \end{pmatrix}. \tag{2.60}$$

However, since P_k is a sub-matrix of P_{k+1} of size $k \times N$ whose rows are formed by the first k rows of P_{k+1}, we get

$$P_k \Delta s^{(k)} = L'_k r_k. \tag{2.61}$$

Now, let $Y_k = P_k^T$ and using the factorization $P_k \Delta S_k = L'_k R_k$, the linear system (2.35) can be written as

$$L'_k R_k \gamma^{(k)} = -P_k \Delta s^{(k)}. \tag{2.62}$$

Using (2.61) and the fact that the matrix L'_k is a nonsingular matrix, (2.62) becomes

$$R_k \gamma^{(k)} = -r_k. \tag{2.63}$$

Since the matrix of this system is upper triangular, the solution is computed simply by back substitution. After getting $\gamma^{(k)}$, $\eta^{(k)}$ is computed from (2.34), and $\xi^{(k)}$ from (2.36). Finally, the approximation $t_{MMPE}^{(k)}$ can be calculated by

$$t_{MMPE}^{(k)} = s^{(0)} + (P_k^T L'_k)(R_k \xi^{(k)}). \tag{2.64}$$

An algorithm for the RRE method is presented in Table (2.4). Note that the null elements of $(l_j)_{p(l)}$ can be replaced by $r_{l,j}$. As a result, the MMPE method requires less storage than the RRE and MPE methods. In the algorithm, the symbol $\leftrightarrow$ indicates the exchange of the data: $x \leftrightarrow y \Leftrightarrow t = x,\ x = y,\ y = t$.

Algorithm of the MMPE method

1. Input: Vectors $s^{(0)}, s^{(1)}, \ldots, s^{(k+1)}$; $p = [1, \cdots, N]$.

2. Let $l^1 = \Delta s_0$;
   Find i_0 such that $|(l^1)_{i_0}| = \|l^1\|_\infty$;
   $p(i) \leftrightarrow p(1)$;
   $r_{11} = (l^1)_{i_0}$; $l^1 = \frac{l^1}{r_{11}}$.

3. For $j = 1, \cdots, k$
   Let $u = \Delta s_j$;
   For $l = 1, \cdots, j$
   $c = (u)_{p(l)}$; $r_{l,j+1} = c$; $(u)_{p(l)} = 0$;
   $(u)_{p(l+1:N)} = (u)_{p(l+1:N)} - c \times (l^j)_{p(l+1:N)}$;
   End
   Determine i_0 such that $|(u)_{i_0}| = \|u\|_\infty$;
   $p(j+1) \leftrightarrow p(i_0)$; $r_{j+1,j+1} = (u)_{i_0}$; $l^{j+1} = u/(u_{i_0})$;
   End

4. Solve the upper triangular linear system:
   $R_k \gamma^{(k)} = -r_k$; $r_k = \left[r_{0,k}, r_{1,k}, \cdots, r_{k-1,k}\right]^T$, $\gamma^{(k)} = \left[\gamma_0^{(k)}, \gamma_1^{(k)}, \cdots, \gamma_{k-1}^{(k)}\right]^T$;
   Let $\gamma_k^{(k)} = 1$ and compute $\lambda = \sum_{i=0}^{k} \gamma_i^{(k)}$;
   Let $\eta_i = (1/\lambda)\, \gamma_i^{(k)}$ for $i = 0, \cdots, k$;

5. Calculate $\xi^{(k)} = \left[\xi_0^{(k)}, \xi_1^{(k)}, \cdots, \xi_{k-1}^{(k)}\right]^T$ by
   $\xi_0^{(k)} = 1 - \eta_0^{(k)}$; $\xi_j^{(k)} = \xi_{j-1}^{(k)} - \eta_j^{(k)}$, $j = 1, \cdots, k-1$, $\xi_{k-1}^{(k)} = \eta_k^{(k)}$;
   Calculate $t_{MMPE}^{(k)}$ by $t_{MMPE}^{(k)} = s^{(0)} + P_k^T L_k' (R_k \xi^{(k)})$

Table 2.4: Algorithm of the MMPE method

2.4.6 Restarted (or cyclic) methods

When applying the algorithms of RRE, MPE, and MMPE (Tables 2.2,2.3 and 2.4) in their complete forms, they become increasingly expensive as k increases, because the work requirement grows quadratically with the number

of iteration steps k. The storage requirement grows linearly with k. To avoid this, these algorithms should be restarted periodically every r steps, for some integer $r > 1$. Below in Table 2.5, a practical strategy of a restarted method is described.

Restarted method every r iterations

For $k = 0$, choose an integer r and an initial vector $s^{(0)}$.

For $k = 1, 2, \ldots,$
  Compute the vectors $s^{(1)}, \cdots, s^{(r)}$.
  Calculate $t^{(r-1)}$ using one of the desired algorithms.
  If $t^{(r-1)}$ satisfies accuracy test, stop.
  Else
    $s^{(0)} = t^{(r-1)}$.
End

Table 2.5: Restarted method every r iterations

2.4.7 Application to linear systems

We will use the description of Sadok and Jbilou in [35]. Consider the system of linear equations

$$Ax = f, \tag{2.65}$$

where $A \in \mathbb{R}^{n\times n}$ is a nonsingular matrix, f and x are vectors of $\mathbb{R}^n$, and x^* denotes the unique solution.

Instead of applying the extrapolation methods directly to (2.65), they will be applied to the preconditioned linear system

$$M^{-1}Ax = M^{-1}f,$$

with a nonsingular matrix M.

If we decompose A in the form

$$A = M - L, \tag{2.66}$$

where M and L are two matrices of order N, then the system (2.65) can be written as

$$x = M^{-1}Lx + M^{-1}f. \tag{2.67}$$

Some methods which deal with this kind of techniques are the Jacobi, Gauss-Seidal and S.O.R methods.

Starting with an initial vector $s^{(0)}$, construct the sequence $\{s^{(j)}\}_j$ by

$$s^{(j+1)} = M^{-1}Ls^{(j)} + M^{-1}f, \quad j = 0, 1, \ldots \tag{2.68}$$

Setting $B = M^{-1}L$ and $b = M^{-1}f$ leads to

$$s^{(j+1)} = Bs^{(j)} + b, \quad j = 0, 1, \ldots \tag{2.69}$$

Note that if the sequence $\{s^{(j)}\}$ is convergent, upon applying the mentioned extrapolation methods on it, all give the exact solution of the linear system (2.65).

Let $C = I - B$, and define the residual $r(x)$ for a vector x by

$$r(x) = b - Cx.$$

From (2.69), we have

$$r(s^{(j)}) = b - Cs_j = s^{(j+1)} - s^{(j)} = \Delta s^{(j)}, \quad \text{the residual of the vector } s^{(j)},$$

and

$$\Delta^2 s^{(j)} = \Delta s^{(j+1)} - \Delta s^{(j)} = -C\Delta s^{(j)},$$

and since $\Delta^2 s^{(k)} = -C\Delta s^{(k)}$, then we have $\Delta^2 S_k = -C\Delta S_k$.

Then it follows from (2.22) and (2.69), that the generalized residual of the approximation $t^{(k)}$ is the true residual

$$\tilde{r}(t^{(k)}) = r(t^{(k)}) = b - Ct^{(k)}. \tag{2.70}$$

Let d be the degree of the minimal polynomial of B for the vector $s^{(0)} - x^*$ and , as C is nonsingular, this polynomial is also the minimal polynomial of B for $r^{(0)} = \Delta s^{(0)}$. Then, the matrices $\Delta S_k = [\Delta s^{(0)}, \ldots, \Delta s^{(k-1)}]$ and $\Delta^2 S_k = [\Delta^2 s^{(0)}, \ldots, \Delta^2 s^{(k-1)}]$ have full rank for $k \leq d$. Also, $t^{(k)}$ exists and is equal to the solution of the linear system (2.65).

The extrapolation methods use implicitly this minimal polynomial and since it is not known in practice, the aim of these methods is to approximate it.

When applied to sequences generated by (2.69), the vector extrapolation methods above produce approximations $t^{(k)}$ such that the corresponding residuals $r^{(k)} = b - Ct^{(k)}$ satisfy the relations

$$r^{(k)} \in W_k = CV_k, \tag{2.71}$$

and

$$r^{(k)} \perp L_k, \tag{2.72}$$

where V_k =span$\{\Delta s^{(0)}, \ldots, \Delta s^{(k-1)}\}$ and L_k depends on the method used:

$$L_k \equiv \begin{cases} W_k\,, & \text{for the RRE} \\ V_k\,, & \text{for the MPE} \\ Y_k = \text{span}\{y_1, \ldots, y_k\}\,, & \text{for the MMPE} \end{cases} \tag{2.73}$$

where $y_1, \ldots, y_k$ are linearly independent vectors.

Remark that, since $W_k \equiv K_k(C, Cr^{(0)})$, the extrapolation methods above are Krylov subspace methods. It is proven in [65] that RRE is an orthogonal projection method and is theoretically equivalent to GMRES while MPE and MMPE are oblique projection methods and are equivalent to the method of Arnoldi and to the Hessenberg method, respectively. from this observation, we conclude that for $k \leq d$, the approximation $t^{(k)}$ exists and is unique, unconditionally for RRE, and this is not always the case for MPE and MMPE. In fact, for the last two methods, the approximation $t^{(k)} (k > d)$ exists if and only if $\det(\Delta S_k^T \Delta^2 S_k) \neq 0$ for MPE and $\det(Y_k^T \Delta^2 S_k) \neq 0$ for MMPE where $Y_k = [y_1, \ldots, y_k]$.

For the following, we will give some comparisons of the residuals generated by these methods. Let P_k be the orthogonal projector onto W_k . Then from (2.71) and (2.72), the residual generated by RRE can be expressed as

$$r_{RRE}^k = r^{(0)} - P_k r^{(0)}. \tag{2.74}$$

We also consider the oblique projectors Q_k and R_k onto W_k and orthogonally to V_k and Y_k respectively. It follows that the residuals produced by MPE and MMPE can be written as

$$r_{MPE}^k = r^{(0)} - Q_k r^{(0)}, \tag{2.75}$$

and

$$r_{MMPE}^k = r^{(0)} - R_k r^{(0)}. \tag{2.76}$$

Denote by θ_k the acute angle between $r^{(0)}$ and the subspace W_k. This angle is defined by

$$\cos\theta_k = \max_{z \in W_k - \{0\}} \left(\frac{|(r^{(0)}, z)|}{\|r^{(0)}\| \|z\|} \right). \tag{2.77}$$

Next, we give some relations satisfied by the residual norms of the three extrapolation methods.

Theorem 2.4.1. *Let ϕ_k be the acute angle between $r^{(0)}$ and $Q_k r^{(0)}$ and let ψ_k denotes the acute angle between $r^{(0)}$ and $R_k r^{(0)}$. Then, we have the following relations:*

1. $\|r_{RRE}^{(k)}\|^2 = (\sin^2\theta_k)\|r^{(0)}\|^2$,
2. $\|r_{MPE}^{(k)}\|^2 = (\tan^2\phi_k)\|r^{(0)}\|^2$,
3. $\|r_{RRE}^{(k)}\| \leq (\cos\phi_k)\|r_{MPE}^{(k)}\|$,
   Moreover if for MMPE $y_j = r^{(0)}$ for some $j = 1,\ldots,k$, then we also have
4. $\|r_{MMPE}^{(k)}\|^2 = (\tan^2\psi_k)\|r^{(0)}\|^2$,
5. $\|r_{RRE}^{(k)}\| \leq (\cos\psi_k)\|r_{MMPE}^{(k)}\|$.

The proof of this theorem can be found in [35]. Note that from relations (1), (2) and (4) of Theorem (2.4.1), we see that the residuals of the RRE are always defined while those produced by MPE and MMPE may not exist. We also observe that if a stagnation occurs in RRE ($\|r_{RRE}^{(k)}\| = \|r^{(0)}\|$ for some $k < d$), then $\cos\theta_k = 0$ and, from (2.77), this implies that $\cos\phi_k = \cos\psi_k = 0$ and hence the approximations produced by MPE and MMPE are not defined.

When the linear process (2.69) is convergent, it is more useful in practice to apply the extrapolation methods after a fixed number p of basic iterations. We note also that, when these methods are used in their complete form, the required work and storage grow linearly with the iteration step. To overcome this drawback we use them in a cycling mode and this means that we have to restart the algorithms after a chosen number m of iterations. This idea will be discussed later.

Stable schemes for the computation of the approximation tk are given in [64, 33]. In [64], Sidi gave an efficient implementation of the MPE and RRE methods which is based on the QR decomposition of the matrix ΔS_k . In [33], Jbilou and Sadok used an LU decomposition of ΔS_k with a pivoting strategy. These implementations require low work and storage and are more stable numerically.

2.4.8 Application to nonlinear systems

Consider the nonlinear system of equations

$$x = G(x), \tag{2.78}$$

where $G : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ and let x^* be a solution of (2.78).

For any arbitrary vector x, the residual is defined by

$$r(x) = G(x) - x.$$

Starting with an initial vector $s^{(0)}$, construct the sequence $\{s^{(j)}\}_j$ by a fixed point iteration,

$$s^{(j+1)} = G(s^{(j)}), \quad j = 0, 1, \ldots \tag{2.79}$$

having s^* as solution, i.e. $\lim_{k\to\infty} s^{(k)} = s^*$

Remarks:

1. $r(s^{(j)}) = \tilde{r}(s^{(j)}) = \Delta s^{(j)}, \quad j = 0, 1, \ldots$

2. Similarly to linear problems, it is more useful to run some basic iterations before the applying one of the extrapolation methods for solving (2.78).

3. As the iteration step k increases, the storage and the evaluation of the function G increase. So, a good way to overcome this problem is by restarting the algorithms after a fixed number of iterations.

4. Extrapolation methods are more efficient if they are applied to a preconditioned nonlinear system $\tilde{G}(x) = x$ where the function $\tilde{G}$ is obtained from G by some preconditioning nonlinear technique.

Below, we give an extrapolation algorithm for solving the nonlinear problem (2.78).

An extrapolation algorithm for a nonlinear system

1. For $k = 0$, choose $x^{(0)}$ and the integers p and m.
2. Basic iteration:
   Set $t^{(0)} = x^{(0)}$;
   $w^{(0)} = t^{(0)}$;
   $w^{(j+1)} = \tilde{G}(w^{(j)}),\ j = 0, \ldots, p-1$.
3. Extrapolation phase:
   $s^{(0)} = w^{(p)}$;
   If $\|s^{(1)} - s^{(0)}\| < \epsilon$, stop;
   Else $s^{(j+1)} = \tilde{G}(s^{(j)}),\ j = 0, \ldots, m$;
   Compute the approximation $t^{(m)}$ by RRE, MPE or MMPE.
4. Set $x^{(0)} = t^{(m)},\ k = k+1$ and go to 2.

Table 2.6: An extrapolation algorithm for a nonlinear system

Similarly to systems of linear equations, efficient computation of the approximation $t^{(m)}$ produced by RRE, MPE and MMPE have been derived in [64, 33]. These implementations give an estimation of the residual norm at each iteration and it allows to stop the algorithms without having to compute the true residual which requires an extra evaluation of the function $\tilde{G}$. Important properties of vector extrapolation methods is the fact that they do not use the knowledge of the Jacobian of the function $\tilde{G}$ and have a quadratic convergence (when they are used in their complete form). Also note that the results of Theorem 2.4.1 are still valid for nonlinear problems by replacing in the relations of this theorem the residual $r^{(k)}$ by the generalized residual $\tilde{r}^{(k)}$. Vector extrapolation methods such as MMPE can also be used for computing eigenelements of a matrix [31].

2.4.9 Quadratic convergence theorem of RRE

This theorem was given by Sadok and Jbilou in [34] as a complete proof of the quadratic convergence of the RRE and MPE methods. Consider the application of the restarted algorithm of these methods of Table 2.5 on the nonlinear system (2.78) with $r = r_k$.

Construct the sequence (2.79) by a fixed-point with an initial approximation $s^{(0)}$ of the solution s^*.

The algorithm which will be considered is the following:

- Choose a starting point $x^{(0)}$.
- At the iteration k, we set $s^{(0)} = x^{(k)}$ and $s^{(i+1)} = G(s^{(i)})$ for $i = 0, \ldots, r_k - 1$, where r_k is the degree of the minimal polynomial of $G'(s^*)$ for the vector $x^{(k)} - s^*$.
- Compute $x^{(k+1)}$ such that $x^{(k+1)} = t^{(r_k)} = s^{(0)} - \Delta S_{k,r_k} \Delta^2 S^+_{k,r_k} \Delta s^{(0)}$.

Theorem (2.4.2) uses the notation

$$\alpha_k(x) = \sqrt{\det(H_k^*(x) H_k(x))}$$

where

$$H_k(x) = \left(\frac{G(x) - x}{\|G(x) - x\|}, \ldots, \frac{G^{r_k}(x) - G^{r_k - 1}(x)}{\|G^{r_k}(x) - G^{r_k - 1}(x)\|} \right)$$

.

Theorem 2.4.2. *Let $J = G'(s^*)$. Suppose that the matrix $J - I$ is regular. Set $M = \|(J - I)^{-1}\|$. The Frechet derivative G' of G satisfies the Lipschitz condition*

$$\|G'(x) - G'(y)\| \leq L\|x - y\|, \ \forall x, y \in \mathbb{D}, \tag{2.80}$$

where $\mathbb{D}$ is an open and convex subset of $\mathbb{C}^n$. If

$$\exists \alpha > 0, \ \exists K, \ \forall k \geq K \ : \ \alpha_k(x^{(k)}) > \alpha, \tag{2.81}$$

then there exists a neighbourhood U of s^* such that for all $s^{(0)} \in U$,

$$\|x^{(k+1)} - s^*\| = O(\|x^{(k)} - s^*\|^2). \tag{2.82}$$

The proof of this theorem can be found in [34].

2.4.10 Operation count and storage

We will use a nice numerical comparison done by Sadok and Jbilou in [35] which describes the operation count (multiplications and additions) and the storage requirements to compute the approximation $t^{(k)}$ with RRE, MPE, and MMPE. Denote by n the dimension of vectors. In practice, n is very large and k is small, so only the main computation work is listed.

Method	RRE	MPE	MMPE
Multiplications and additions	$2nk^2$	$2nk^2$	nk^2
Memory storage	$(k+1)n$	$(k+1)n$	$(k+1)n$

Table 2.7: Computational costs and memory requirements for RRE, MPE, and MMPE.

The implementations of RRE, MPE, and MMPE given in [33, 64] allows to compute exactly the norm of the residual at each iteration for linear systems or to estimate it for nonlinear systems without actually computing the residuals. This reduces the cost of implementation and is used to stop the algorithms when the accuracy is achieved.

2.4.11 Remarks on algorithms for extrapolation methods

An important issue concerning the extrapolation methods is the development of efficient algorithms for implementing existing extrapolation methods. An efficient algorithm is one that involves a small number of arithmetic operations and little storage when storage becomes a problem.

Convergence and stability

The analysis of convergence and stability is the most important subject in the theory of extrapolation methods. It is also the richest in terms of the variety of results that exist and still can be obtained for different extrapolation methods and sequences. Thus, it is impossible to make any specific remarks about convergence and stability. However, several remarks on the approach to these topics can be made. We start with the topic of convergence analysis.

Study of convergence

First step in the convergence analysis of extrapolation methods is the formulation of the conditions that we impose on a sequence $\{s^{(k)}\}$. In this thesis, we deal with sequences that arise in common applications. Therefore, we emphasize on the conditions that are relevant to these applications. The number of these conditions are kept to a minimum which leads to mathematically more valuable results.

Next, the analysis of the errors $\{t^{(k)}\} - \{s^{(k)}\}$ under these conditions, where $\{t^{(k)}\}$ is the extrapolated sequence, is discussed. This analysis leads to different

types of results depending on the complexity of the situation. In some cases, we obtain only the most dominant term of this expansion of $\{t^{(k)}\} - \{s^{(k)}\}$ for $k \to \infty$. While in others, we obtain a realistic upper bound on $|\{t^{(k)}\} - \{s^{(k)}\}|$ from which powerful convergence results can be obtained.

An important feature, which can be found in [66], is that it is not only feasible to show that the sequence $\{t^{(k)}\}$ converges faster than $\{s^{(k)}\}$, but instead it is also possible to obtain the precise asymptotic behavior of the corresponding acceleration factor or a good upper bound for it.

Study of stability

When we compute the sequence $\{t^{(k)}\}$ in finite-precision arithmetic, we obtain a sequence $\{\bar{t}^{(k)}\}$ that is different than $\{t^{(k)}\}$, the exact transformed sequence. This is caused mainly by errors (roundoff errors and errors of other kinds as well) in the $s^{(k)}$. Normally, we would like to know by how much $\{\bar{t}^{(k)}\}$ differs from $\{t^{(k)}\}$. That is, we want to estimate $|\bar{t}^{(k)} - t^{(k)}|$. This is important since the knowledge of $|\bar{t}^{(k)} - t^{(k)}|$ helps in assessing the cumulative error $|\bar{t}^{(k)} - s|$ in $\bar{t}^{(k)}$; see [66].

It is possible to achieve sufficient accuracy in $\bar{t}^{(k)}$ by computing the $s^{(k)}$ with high accuracy. This can be accomplished by on a computer by doubling the precision of the floating-point arithmetic used for computing the $s^{(k)}$. When applying an extrapolation method to a convergent sequence $\{s^{(k)}\}$ numerically, we would like to be able to compute the sequence $\bar{t}^{(k)}$ without $|\bar{t}^{(k)} - t^{(k)}|$ becoming unbounded for increasing k.

Conclusion

In chapter 2 of this thesis, we explained what is meant by extrapolation methods and the theory of extrapolation.

Two categories of extrapolation methods were presented: the scalar extrapolation and the vector extrapolation. A description of the most popular methods of both types was presented with emphasizing on the vector type, and in particular, the polynomial vector extrapolation methods.

After the definition of each polynomial method, a suitable implementation and algorithm were proposed. The application of these methods to linear and nonlinear systems of equations was discussed. Also remarks on these algorithms, their convergence, and stability were noted.

Algebraic Riccati Equations Arising in Transport Theory (NARE)

CHAPTER 3

Algebraic Riccati Equations Arising in Transport Theory (NARE)

3.1 Introduction to (NARE)

In 1723, Riccati equations were first studied by Jacopo Francesco Riccati from whom these equations got their name. Individual cases of these equations were examined by D. Bernoulli (1724-1725), J. Liouville (1841), and still being studied till nowadays. The matrix version of these equations came much later.

From now on, the term "Riccati equation" will be used to refer to matrix equations with an analogous quadratic term, which occur in both continuous-time and discrete-time linear-quadratic-Gaussian control. The steady-state (non-dynamic) version of these is referred to as the algebraic Riccati equation.

Nonsymmetric algebraic Riccati equations (NARE) are quadratic matrix equations of the general form:

$$XCX - XD - AX + B = 0, \tag{3.1}$$

where A, B, C, and D are real matrices. The square matrix X is the unknown.

The term *nonsymmetric* distinguishes this case from the widely studied continuous-time algebraic Riccati equations, defined by the quadratic matrix equation $XCX - AX - XA^T + B = 0$, where B and C are symmetric. We refer the reader to the books [43, 47] for a comprehensive analysis of continuous-time algebraic Riccati equations.

The matrix coefficients of the (NARE) (1) define the $n \times n$ M-matrix

$$M = \begin{bmatrix} D & -C \\ -B & A \end{bmatrix}. \tag{3.2}$$

This assumption is motivated by the increasing applicative interest of this kind of NAREs, and by the recent theoretical and algorithmic advances that

have been achieved. This thesis concerns algebraic Riccati equations associated with nonsingular or singular irreducible M-matrices. The case in which M is singular and reducible is of minor interest.

We recall that a real square matrix M is an M-matrix if it can be written as $M = sI - N$ with $N \geq 0$. M is said to be a nonsingular M-matrix if $s > \rho(N)$, where $\rho(N)$ is the spectral radius of N. We say that equation (3.1) is associated with an M-matrix if its coefficients form an M-matrix M.

Nonsymmetric algebraic Riccati equations arise in transport theory, the Wiener-Hopf factorization of Markov chains, nuclear physics, applied probability, engineering, network theory, control theory and in so many fields. There are two important applications where nonsymmetric algebraic Riccati equations associated with M-matrices play a very important role: the study of fluid queues models [54, 55, 73], and the analysis of transport equations [38, 40]. In both cases the solution of interest is the matrix X^* with positive entries, which among all the positive solutions is the one with componentwise minimal entries. We call any solution X^* sharing this property a minimal positive solution.

In this thesis, we are interested in a special case of Riccati equations arising in transport theory with A, B, C, and D are given specific values. When we talk about this kind of Riccati equations arising in transport theory , this means that we are dealing with particle transport (or radiative transfer).

3.2 Existence of nonnegative solutions

The existence of nonnegative solutions of (3.1) was illustrated by the degree theory of Juang [37]. It is shown [40, 37] that the Riccati equation (3.7) has two entry-wise positive solutions X=$[x_{ij}]$ and Y=$[y_{ij}] \in \mathbb{R}^{n\times n}$ which satisfies $X \leq Y$, where we use the notation that $X \leq Y$ if $x_{ij} \leq y_{ij}$ for all $i, j = 1, ..., n$.

In the applications from transport theory only X, the smaller one of the two positive solutions is of interest and is physically meaningful. Some iterative procedures [39] were developed to find these nonnegative solutions. However, their convergence rates are very slow which is unsatisfactory.

In fact, the special structure of the matrix M of (3.17) allows one to prove the existence of a minimal nonnegative solution X^* of (4.6), i.e., a solution $X^* > 0$ such that $X - X^* > 0$ for any solution $X > 0$ to (4.6). See [21, 23] for more details.

Theorem 3.2.1. *Let M in* (4.1.2) *be an M-matrix. Then the NARE* (4.6) *has a minimal nonnegative solution X^*. If M is irreducible, then $X^* > 0$ and $A - X^*C$ and $D - CX^*$ are irreducible M-matrices. If M is nonsingular, then $A - X^*C$ and $D - CX^*$ are nonsingular M-matrices.*

3.3 Matrix form of NARE

Nonsymmetric algebraic Riccati equations appear in transport theory, a variation of the usual one-group neutron transport equation [4, 11, 18], where the mathematical model consists in solving the integrodifferential equation

$$\left(\frac{1}{\mu+\alpha}+\frac{1}{y-\alpha}\right)L(\mu,y)=c\left(1+\frac{1}{2}\int_{-\alpha}^{1}\frac{L(\omega,y)}{\omega+\alpha}d\omega\right)\left(1+\frac{1}{2}\int_{\alpha}^{1}\frac{L(\mu,\omega)}{\omega-\alpha}d\omega\right), \quad (3.3)$$

with $(\mu,y)\in[-\alpha,1]\times[\alpha,1]$ and $(\alpha,c)\in[0,1]$ are given nonnegative constants; see appendix in [41]. Here, the function $L:[-\alpha,1]\times[\alpha,1]\to\mathbb{R}$, a real valued function, is called the scattering function.

This variation is formulated as

$$\left[(\mu+\alpha)\frac{\partial}{\partial x}+1\right]\varphi(x,\mu)=\frac{c}{2}\int_{-1}^{1}\varphi(x,w)\,dw\,,$$
$$\varphi(0,\mu)=f(\mu)\,,\qquad \mu>-\alpha\,,\ |\mu|\le 1\,,$$
$$\lim_{x\to\infty}\varphi(x,\mu)=0\,,$$

Here, φ is the neutron flux, α $(0\le\alpha<1)$ is an angular shift, and c is the average of the total number of particles emerging from a collision, which is assumed to be conserved, i.e., $0<c\le 1$. The case where $c=0$ or $\alpha=1$, the integrodifferential equation (3.3) has a trivial solution [4]. However, when $0\le\alpha<1$ and $0<c\le 1$, it has a unique, positive, uniformly bounded, and globally defined solution [41].

Remark 3.3.1. *Discretization of the integrodifferential equation (3.3) by a numerical quadrature formula on the interval* $[0,1]$ *yields an algebraic matrix Riccati equation, see the following for more details.*

Denote by $\{\omega_k\}_{k=1}^{n}$ and $\{c_k\}_{k=1}^{n}$ the sets of weights nodes and weights, respectively, of the specific quadrature rule that is used on the interval [0, 1], which is obtained by dividing the interval into n/4 subintervals of equal length and applying a Gauss-Legendre quadrature with 4 nodes to each subinterval.

These typically satisfy:

$$\begin{cases} c_1,\ldots,c_n>0, \qquad \displaystyle\sum_{k=1}^{n}c_k=1\,,\\ 1>\omega_1>\omega_2>\ldots>\omega_n>0\,. \end{cases} \quad (3.4)$$

Upon transforming the nodes and weights of the quadrature rule (Gauss-Legendre) on [0 , 1] to the intervals $[-\alpha\ ,\ 1]$ and $[\alpha\ ,\ 1]$, respectively, we have the following relationships:

$$\begin{cases} \omega_k^- = \{(1+\alpha)\omega_k - \alpha\} \ \in [-\alpha\ ,\ 1], \\ \omega_k^+ = \{(1-\alpha)\omega_k + \alpha\} \ \in [\ \alpha\ ,\ 1], \\ c_k^- = c_k(1+\alpha), \\ c_k^+ = c_k(1-\alpha), \end{cases} \tag{3.5}$$

for $k = 1\ ,\ \ldots,\ n$.

Let $T_{ij} = T(\omega_i^-\ ,\ \omega_j^+)$, for $i, j\ =\ 1,\ \ldots\ ,\ n$. Replacing $x,\ y$ with ω_i^- and ω_j^+, respectively, in 3.3, the integrals in 3.3 can be approximated by:

$$\int_{-\alpha}^{1} \frac{T(\omega\ ,\omega_j^+)}{\omega+\alpha}\, dw\ \sim \sum_{k=1}^{n} \frac{c_k^- T_{kj}}{\omega_k^- + \alpha}$$

and

$$\int_{\alpha}^{1} \frac{T(\omega_i^-,\omega)}{\omega-\alpha}\, dw\ \sim \sum_{k=1}^{n} \frac{c_k^+ T_{ik}}{\omega_k^+ - \alpha}$$

Consequently, the descretized version of 3.3 becomes:

$\frac{1}{c(\omega_i^-+\alpha)} T_{ij}\ +\ \frac{1}{c(\omega_j^+-\alpha)} T_{ij} =$

$$1 + \frac{1}{2}\sum_{k=1}^{n} \frac{c_k^- T_{kj}}{\omega_k^- + \alpha}\ +\ \frac{1}{2}\sum_{k=1}^{n} \frac{c_k^+ T_{ik}}{\omega_k^+ - \alpha}\ +\ \frac{1}{4}\sum_{k=1}^{n}\sum_{l=1}^{n} \frac{T_{ik} c_k^+ c_l^- T_{ij}}{(\omega_k^+ - \alpha)(\omega_l^- + \alpha)}. \tag{3.6}$$

Substituting the values of $w_k^+, w_k^-,\ c_k^+$, and c_k^- into 3.6 and writing the resulting equation in matrix form, we get an $n \times n$ nonsymmetric algebraic matrix Riccati equation of the general form

$$XCX\ -\ XD\ -\ AX\ +\ B\ =0 \tag{3.7}$$

for a real square unkown matrix X and where A, B, C, and $D \in \mathbb{R}^{n\times n}$ are given by

$$A = \Delta - eq^T,\ B = ee^T,\ C = qq^T, \text{ and } D = \Gamma - qe^T, \tag{3.8}$$

Here,

$$\left.\begin{array}{lll} e = [1,1,...,1]^T, & & \\ q = [q_1, q_2, ..., q_n]^T & \text{with} & q_i = \frac{c_i}{2\omega_i}, q_i > 0, \\ \Delta = \text{diag}([\delta_1, \delta_2, ..., \delta_n]) & \text{with} & \delta_i = \frac{1}{c\omega_i(1+\alpha)}, \\ \Gamma = \text{diag}([\gamma_1, \gamma_2, ..., \gamma_n]) & \text{with} & \gamma_i = \frac{1}{c\omega_i(1-\alpha)}. \end{array}\right\} \tag{3.9}$$

where $0 \le \alpha < 1$, and $0 < c \le 1$.

Again, $\{\omega_i\}_{i=1}^n$ and $\{c_i\}_{i=1}^n$ satisfies conditions (3.4).
Consequently, we have

$$0 < \delta_1 < \delta_2 < \ldots \delta_n,$$

and

$$0 < \gamma_1 < \gamma_2 < \ldots < \gamma_n.$$

In addition,

$$\begin{cases} \gamma_i = \delta_i; & \text{for } \alpha = 0, \\ \gamma_i > \delta_i; & \text{for } \alpha \neq 0. \end{cases} \quad \text{for } i = 1, 2, \ldots, n$$

Note that when $\alpha = 0$, the nonsymmetric equation (3.7) turns to a symmetric algebraic Riccati equation.

3.4 The solution of NARE

It has been shown by Lu [46] that the minimal positive solution of NARE can be obtained via computing the minimal positive solution of a vector equation, which is derived from the special form of solutions of the Riccati equation and by exploitation of the special structure of the coefficient matrices of the Riccati equation.Also in [46], a simple iteration was developed to compute the minimal positive solution of this vector equation. Bao, Lin, and Wei [3] suggested a modified version of this simple iteration which succeeded to be more efficient. However, Newton's method was proposed in [45] to solve the vector equation. But it requires one LU factorization [20], for a matrix of order n, at each iteration.

Solution form and the simple iteration

Rewrite NARE (3.7) as

$$\Delta X + XD = (Xq + e)(q^T X + e^T), \tag{3.10}$$

and let

$$u = Xq + e \quad \text{and} \quad v^T = q^T X + e^T. \tag{3.11}$$

It has been shown in [40, 37] that any solution of (3.7) must be of the form

$$X = T \circ (uv^T) = (uv^T) \circ T, \tag{3.12}$$

where $\circ$ denotes Hadamard product defined by $A \circ B = [a_{ij}.b_{ij}]$ for any two matrices $A = [a_{ij}]$ and $B = [b_{ij}]$, T is the Cauchy matrix, $T = [t_{i,j}] = [1/(\delta_i + \gamma_j)]$,

$X = (x_{i,j})$, $u = (u_1, u_2, \ldots, u_n)^T$ and $v = (v_1, v_2, \ldots, v_n)^T$.

Finding the minimal positive solution of (3.7) requires finding proper positive vectors u and v in (3.12). Substituting (3.12) in (3.11) leads to the following vector equation

$$\begin{cases} u = u \circ (Pv) + e, \\ v = v \circ (Qu) + e, \end{cases} \tag{3.13}$$

where,

$$P = [P_{ij}] = [\frac{q_j}{\delta_i + \gamma_j}] = T.\mathrm{diag}(q), \tag{3.14}$$

and

$$Q = [Q_{ij}] = [\frac{q_j}{\delta_j + \gamma_i}] = T^T.\mathrm{diag}(q). \tag{3.15}$$

Then, the minimal positive solution of (3.7) can be obtained via computing the minimal positive solution of the vector equation (3.13).

The vector equation (3.13) can be reformulated as

$$\begin{cases} f(u,v) := u - u \circ (Pv) - e = 0, \\ g(u,v) := v - v \circ (Qu) - e = 0. \end{cases} \tag{3.16}$$

which is a system of nonlinear equations having the vector pair (u, v) as a positive solution. Based on the above, Lu [46] defined a simple iteration of (3.13).

3.5 Main iterative methods to find the minimal positive solution of NARE

The equation of NARE (3.7)-(3.8) was shown, by Guo in [23], to fall in the class of NAREs associated with a nonsingular M-matrix or a singular irreducible M-matrix. In fact, arranging the coefficients as

$$M = \begin{bmatrix} D & -C \\ -B & A \end{bmatrix}. \tag{3.17}$$

yields an M-matrix.

For this class of AREs, several suitable algorithms exist for computing the minimal positive solution: Newton method [26], the logarithmic and cyclic reduction [7, 22], and the structure-preserving doubling algorithm [25, 28]. All these methods share the same order of complexity, that is $O(n^3)$ arithmetic ops per step, and provide quadratic convergence in the generic case.

Observing that NARE (3.7) on Page 54 is defined by a linear number of parameters, it is normal to aim to design algorithms which exploit the structure of the matrices and thus have a cost of order lower than $O(n^3)$ ops.

In this direction, Lu [46] designed an iteration of a vector equation whose minimal positive solution leads to recovering the minimal positive solution of NARE.

3.5.1 The Iterative Method of Lu

Lu defined an iterative scheme to find the positive vectors u and v,

$$\begin{cases} u^{(k+1)} = u^{(k)} \circ (Pv^{(k)}) + e, \\ v^{(k+1)} = v^{(k)} \circ (Qu^{(k)}) + e, \text{ for } k = 0,1,\ldots \\ (u^{(0)}, v^{(0)}) = (0,0) \end{cases} \tag{3.18}$$

satisfying a certain stopping criterion.

For all $0 \leq \alpha < 1$ and $0 < c \leq 1$, the sequence $\{(u^{(k)}, v^{(k)})\}$ defined by (3.18) is strictly monotonically increasing, bounded above, and thus converging; see [46] for the proof. Each iteration costs about $4n^2$ flops.

The minimal positive solution of (3.7) can be computed by

$$X^* = T \circ (u^* (v^*)^T), \tag{3.19}$$

where (u^*, v^*) is the limit of $(u^{(k)}, v^{(k)})$ defined by the iteration scheme (3.18).

The iteration of Lu (3.18) has a computational cost of $O(n^2)$ ops per step and converges linearly for $\alpha \neq 0$ or $c \neq 1$. The linear convergence is a drawback since the algorithm in many cases needs a large number of iterations to converge and it is outperformed by algorithms with quadratic convergence and $O(n^3)$ ops.

3.5.2 A Modified Iterative Method

Bao et al. [3] proposed a modified version of the iterative method of Lu [46]. They noticed that as $u^{(k+1)}$ is obtained before $v^{(k+1)}$, it should be a better approximation to u^* than $u^{(k)}$. Consequently, $u^{(k)}$ is replaced with $u^{(k+1)}$ in the equation for $v^{(k+1)}$ of the iteration scheme (3.18).

The modified iteration is as follows

$$\begin{cases} \tilde{u}^{(k+1)} = \tilde{u}^{(k)} \circ (P\tilde{v}^{(k)}) + e \ , \\ \tilde{v}^{(k+1)} = \tilde{v}^{(k)} \circ (Q\tilde{u}^{(k+1)}) + e \ , \text{ for } k = 0,1,\ldots \\ \tilde{u}^{(0)} = \tilde{v}^{(0)} = 0. \end{cases} \tag{3.20}$$

The monotonic convergence of the modified iteration scheme (3.20) is illustrated in [3]. Also, it is shown how the minimal positive solution X^* of the NARE (3.7) can be computed from

$$X^* = T \circ (u^*(v^*)^T),$$

where (u^*, v^*) is the limit of $(\tilde{u}^{(k)}, \tilde{v}^{(k)})$ defined by the modified iteration (3.20).

Since it was also shown that $u^{(k)} < \tilde{u}^{(k)}$ and $v^{(k)} < \tilde{v}^{(k)}$ for $k \geq 3$, $(\tilde{u}^{(k)}, \tilde{v}^{(k)})$ has the same limit (u^*, v^*) as $(u^{(k)}, v^{(k)})$. Also, this proves how the modified iteration scheme (3.20) is more efficient than (3.18). The cost of every iteration step is about $4n^2$ flops.

The Nonlinear Splitting Iteration Methods

Two accelerated variants of the iterative scheme (3.18) of Lu, proposed in [2], are the nonlinear block Jacobi (NBJ) iteration scheme

$$\begin{cases} u^{(k+1)} = u^{(k+1)} \circ (Pv^{(k)}) + e, \\ v^{(k+1)} = v^{(k+1)} \circ (Qu^{(k)}) + e, \text{ for } k = 0, 1, \ldots, \end{cases} \tag{3.21}$$

and the nonlinear block Gauss-Seidel (NBGS) iteration scheme

$$\begin{cases} u^{(k+1)} = u^{(k+1)} \circ (Pv^{(k)}) + e, \\ v^{(k+1)} = v^{(k+1)} \circ (Qu^{(k+1)}) + e, \text{ for } k = 0, 1, \ldots, \end{cases} \tag{3.22}$$

For both NBJ and NBGS, the sequence $\{(u^{(k)}, v^{(k)})\}$ is shown in [2] to be strictly monotically increasing and convergent to the minimal positive solution (u^*, v^*) of the vector equations (3.13). NBJ and NBGS have the same computational costs at every iteration step (about $4n^2$ flops). Both are effective solvers of NAREs arising in transport theory but NBGS is more efficient than NBJ in applications. In particular, in terms of the asymptotic rates of convergence, the NBGS method is twice as fast as the NBJ method; see Theorem 5 in [27] which explains the numerical results presented in [2], where the number of iterations required for NBGS is half of that for NBJ.

Note that the four fixed-point iterations (3.18), (3.20), (3.21), and (3.22) are easy to use and share the same low complexity at every iteration. However, NBGS was proved in [27] to be the fastest among these methods in terms of the asymptotic rate of convergence when $(\alpha, c) \neq (0, 1)$, although being sublinear when $(\alpha, c) = (0, 1)$. The sublinear convergence, which takes place when the Jacobian at the required solution is singular, is transformed into a quadratic convergence by means of a Newton method proposed in [8].

3.5.3 The Newton Method

Newton's iteration was first applied to the symmetric algebraic Riccati equation by Kleinman in 1968 [42] and later on by various authors. In particular, Benner and Byers [5] complemented the method with an optimization technique in order to reduce the number of steps needed for arriving at convergence. The study of the Newton method for nonsymmetric algebraic Riccati equations was started by Guo and Laub in [26] who proposed a method which has an order of complexity $O(n^3)$ ops per step and yields quadratic convergence. However, a nice convergence result was given by Guo and Higham in [24].

The convergence of the Newton method is generally quadratic except for the critical case where the convergence is observed to be linear with rate 1/2 [24]. At each step, a Sylvester matrix equation must be solved, so the computational cost is $O(n^3)$ ops per step, but with a large overhead constant.

Newton's iteration applied to NARE (3.7) on Page 54, for a suitable initial value $X^{(0)}$, generates the matrix sequence $\{X^{(k)}\}$ defined by the solution of the Sylvester equation

$$(X^{(k+1)} - X^{(k)})(D - CX^{(k)}) + (A - X^{(k)}C)(X^{(k+1)} - X^{(k)}) = R(X^{(k)}) \tag{3.23}$$

where $R(X) = XCX - XD - AX + B$ is the Riccati operator.

Using the Kronecker product notation, equation (3.23) can be rewritten as

$$\mathrm{vec}X^{(k+1)} - \mathrm{vec}X^{(k)} = ((D - CX^{(k)})^T \otimes I_n + I_n \otimes (A - X^{(k)}C))^{-1}\mathrm{vec}R(X^{(k)}), \tag{3.24}$$

where the vec operator stacks the columns of a matrix one above the other to form a single vector.

Thus, Newton's iteration is well defined when the matrix

$$J_{X^{(k)}} = (D - CX^{(k)})^T \otimes I_n + I_n \otimes (A - X^{(k)}C) \tag{3.25}$$

is nonsingular for each k. For simplicity, call $J_{X^{(k)}}$ the Jacobian matrix at X.

A sufficient condition for the convergence of the Newton's method is given by Guo and Higham [24] through the following theorem.

Theorem 3.5.1. *Let M* (3.2) *be a nonsingular M-matrix or an irreducible singular M-matrix and let $X^{(0)} = 0$. Then, Newton's iteration* (3.23) *is well defined, and the sequence $\{X^{(k)}\}$ converges monotically to the minimal positive solution of the NARE* (3.7). *Moreover, the Jacobian matrix* (3.25) *$J_{X^{(k)}} \in \mathbb{R}^{n^2 \times n^2}$ is a nonsingular M-matrix for all $k \geq 0$.*

On the other hand, Lu [45] proposed an algorithm that consists of iteration (3.18) combined with Newton iteration to speed up the computation. The algorithm starts with the linear iteration of complexity $O(n^2)$ and switches to a quadratically convergent one, of complexity $O(n^3)$, when some conditions are satisfied. This scheme is simple and more efficient than applying the Newton method directly to (3.7).

Another approach was proposed by Bini et al. [8] who apply a fast Newton method to NARE (3.7). Below, more details concerning this method are presented.

A Fast Newton Method

The fast Newton method of Bini et al. [8] is an efficient method which solves NARE, and thus it would be interesting to compare our results with it. This shall be done in the numerical experiments of Chapter 4.

This method consists of an algorithm which performs the Newton iteration in $O(n^2)$ ops. Considering the customary Newton method applied to NARE (3.7) and exploiting the rank structure of the matrix coefficients leads to an algorithm for performing Newton step in $O(n^2)$ ops. This approach relies on a modification of the fast LU factorization algorithm for Cauchy-like matrices proposed by Gohberg et al. in [19]. The same idea reduces the cost of Newton's algorithm proposed by Lu [45] from $O(n^3)$ to $O(n^2)$ ops while preserving the quadratic convergence in the generic case. Bini, Inannazo, and Poloni considered in [8] a more eneral case when the matrix M is a generic diagonal plus rank-one matrix. Hence (3.8) becomes

$$A = \Delta - \tilde{e}q^T, \quad B = \tilde{e}e^T$$

$$C = \tilde{q}q^T, \quad D = \Gamma - \tilde{q}e^T,$$

where$e, q, \tilde{e}, \tilde{q}$ are any nonnegative vectors such that M, as defined in Theorem 3.5.1, is a nonsingular M-matrix or a singular irreducible M-matrix.

By exploiting the rank structure of the matrix coefficients, an algorithm which relies on a fast LU factorization is proposed for implementing the Newton step for the solution of the system

$$\begin{cases} R^{(k)}x = b, \\ R^{(k)} = I_{2n} - V\mathbb{D}^{-1}U^{(k)} \end{cases} \tag{3.26}$$

in $O(n^2)$ ops, where

$$\mathbb{D} = \Gamma^T \otimes I_n + I_n \otimes \Delta,$$
$$U^{(k)} = [v^{(k)} \otimes I_n \;\; I_n \otimes u^{(k)}]$$
$$\text{(with } u^{(k)} = \tilde{e} + X^{(k)}\tilde{q};\; v^{(k)} = e + X^{(k)T} q),$$
$$V = \begin{bmatrix} \tilde{q}^T \otimes I_n \\ I_n \otimes q^T \end{bmatrix}.$$

This algorithm relies on a suitable modification of the fast LU factorization algorithm for Cauchy-like matrices proposed by Gohberg, Kailath, and Olshevsky in [19].

Algorithm of the Fast Newton's step

function $X^{(k+1)} =$ **NewtonStep** $(X^{(k)})$
$u^{(k)} = \tilde{e} + X^{(k)}\tilde{q}$;
$v^{(k)} = e + X^{(k)T} q$;
$R(X^{(k)}) = u^{(k)} v^{(k)T} - X^{(k)}\Gamma - \Delta X^{(k)}$;
$R_1 = [\tilde{q}^T \otimes I_n; I_n \otimes q^T](\mathbb{D}^{-1} * vec(R(X^{(k)})))$;
$R_2 =$ fastsolve $((I_{2n} - V\mathbb{D}^{-1}U^{(k)})R_2 = R_1)$;
$X^{(k+1)} = \mathbb{D}^{-1}(vec(R(X^{(k)})) + [v^{(k)} \otimes I_n \quad I_n \otimes u^{(k)}] * R_2)$;
return $X^{(k+1)}$
end function

Table 3.1: Algorithm of the Fast Newton's step

Here, the function **fastsolve** implements the fast LU factorization algorithm for Cauchy-like matrices. It includes partial pivoting and is complemented with back-substitution with complexity $O(n^2)$ that was used in the algorithm of Table 3.1. See [8] for more details about the function **fastsolve**.

3.5.4 The Iterative Method of Lin

This method was proposed in [44]. Let $w = [u^T, v^T]^T$ and reformulate the vector equation (3.13) as

$$f(w) := w - w \circ Lw - e = 0, \tag{3.27}$$

where,

$$L = \begin{bmatrix} 0 & P \\ Q & 0 \end{bmatrix}.$$

The Jacobian matrix $J(f,w)$ of $f(w)$ for any $w \in \mathbb{R}^{2n}$ is given by

$$J(f,w) = I_{2n} - N(w);$$

where

$$N(w) = \begin{bmatrix} \text{diag}(Pv) & \text{diag}(u)P \\ \text{diag}(v)Q & \text{diag}(Qu) \end{bmatrix}.$$

Lin constructed a class of iterative methods to solve the vector equation (3.27) based on the following iterative scheme

$$w^{(k+1)} := w^{(k)} - T_k^{-1} f(w^{(k)}), \quad \text{for } k = 0,1,2,\ldots \tag{3.28}$$

where T_k is chosen to approximate $J(f,w^{(k)})$ and $T_k \geq J(f,w^{(k)})$. Note that when $T_k = J(f,w^{(k)})$, the Newton method results. A disadvantage of the Newton method is that it requires an LU factorization of the Jacobian matrix $J(f,w^{(k)})$ at each iteration step to obtain the new approximation to w^*. This costs $O(n^3)$ operations per step.

For any $w^{(k)} = [(u^{(k)})^T, (v^{(k)})^T]^T \in \mathbb{R}^{2n}$, Lin [44] proposed the choice

$$T_k = I_{2n} - \begin{bmatrix} \text{diag}(Pv^{(k)}) & 0 \\ 0 & \text{diag}(Qu^{(k)}) \end{bmatrix}. \tag{3.29}$$

Substituting (3.29) into (3.28) gives

$$w^{(k+1)} = \begin{bmatrix} u^{(k+1)} \\ v^{(k+1)} \end{bmatrix} := \begin{bmatrix} (I_n - \text{diag}(Pv^{(k)}))^{-1} e \\ (I_n - \text{diag}(Qu^{(k)}))^{-1} e \end{bmatrix}. \tag{3.30}$$

Convergence of the iterative method (3.30) was examined in [44]. It was shown that the convergence is sublinear as (α, c) tends to $(0,1)$, and linear when $(\alpha,c) = (0,1)$. Since $(I_n - \text{diag}(Pv^{(k)}))$ and $(I_n - \text{diag}(Qu^{(k)}))$ are diagonal matrices, the computational cost of each iteration step is about $4n^2$ flops. A numerical comparison shows how the iterative method of Lin has a faster convergence than the modified iterative method of Lu (3.20) and the Newton method.

3.5.5 A Modification of the Iterative Method of Lin

Idea of the modification

A modification of the iterative scheme (3.30) that is an analogue of the NBGS method [2] is proposed. The iterative scheme (3.30) computes $u^{(k+1)}$ before

$v^{(k+1)}$. Therefore, $u^{(k+1)}$ should be a better approximation of u^* than $u^{(k)}$. Replacing $u^{(k)}$ by $u^{(k+1)}$ in the equation for $v^{(k+1)}$ in (3.30) leads to the modified iteration scheme

$$w^{(k+1)} = \begin{bmatrix} u^{(k+1)} \\ v^{(k+1)} \end{bmatrix} := \begin{bmatrix} (I_n - \text{diag}(Pv^{(k)}))^{-1}e \\ (I_n - \text{diag}(Qu^{(k+1)}))^{-1}e \end{bmatrix}. \tag{3.31}$$

For a matrix $G \in \mathbb{R}^{n\times n}$, we define the following function

$$\begin{aligned} \phi_G : \quad \mathbb{R}^n &\longrightarrow \mathbb{R}^n, \\ t &\longrightarrow (I_n - \text{diag}(Gt))^{-1}e \end{aligned}$$

Then iteration (3.31) can be expressed as

$$w^{(k+1)} = \begin{bmatrix} u^{(k+1)} \\ v^{(k+1)} \end{bmatrix} := \begin{bmatrix} \phi_P(v^{(k)}) \\ \phi_Q(u^{(k+1)}) \end{bmatrix}. \tag{3.32}$$

Remark 3.5.1. *See Table 4.6 on Page 82 and Figure 4.9 on Page 82 to examine the efficiency of the approach of modification.*

Lemma 1. *Let* (u^*, v^*) *be the minimal positive solution of* (3.31). *Then,* $u^* > e$ *and* $v^* > e$.

Proof. Let (u^*, v^*) be the minimal positive solution of (3.31). Then

$$\begin{cases} u^* = (I_n - \text{diag}(Pv^*))^{-1}e > e > 0 \ , \\ v^* = (I_n - \text{diag}(Qu^*))^{-1}e > e > 0 \ , \end{cases} \tag{3.33}$$

since P and Q are positive matrices. □

Theorem 3.5.2. *Given the sequence of vectors generated by* (3.31) *with initial vector* $(u^{(0)}, v^{(0)}) = (0,0)$ *and* (u^*, v^*) *the minimal positive solution of* (3.31). *Then the sequence* $\{(u^{(k)}, v^{(k)})\}$ *is strictly monotically increasing, bounded above, and thus converging.*

Proof. To prove this theorem, we have to prove componentwise that

(i) $0 \le u^{(k)} < u^{(k+1)} < u^*$ and $0 \le v^{(k)} < v^{(k+1)} < v^*$, $k \ge 1$;

(ii) $\lim_{k\to\infty} u^{(k)} = u^*$ and $\lim_{k\to\infty} v^{(k)} = v^*$.

We start by proving (i) by induction. Let $(u^{(0)}, v^{(0)}) = (0,0)$ be a starting point for this method. For $k = 0$, using (3.31) and Lemma 1, we get

$$u^{(0)} = 0 < e = u^{(1)} < u^*.$$

Also,

$$\begin{aligned} v^{(1)} = [I_n - \mathrm{diag}(Qu^{(1)})]^{-1} e &= [I_n - \mathrm{diag}(Qe)]^{-1} e \\ &< [I_n - \mathrm{diag}(Qu^*)]^{-1} e = v^*, \end{aligned}$$

then,

$$v^{(0)} = 0 < v^{(1)} < v^*.$$

This shows that (i) holds for $k = 0$. Now, suppose that (i) holds for a positive integer k, i.e., we have

$$0 \le u^{(k)} < u^{(k+1)} < u^* \qquad \text{and} \qquad 0 \le v^{(k)} < v^{(k+1)} < v^*, \text{ for } k \ge 1.$$

Using (3.31) and the fact that $v^{(k)} < v^{(k+1)}$, we have

$$\begin{aligned} [I_n - \mathrm{diag}(Pv^{(k+1)})]u^{(k+2)} = e &= [I_n - \mathrm{diag}(Pv^{(k)})]u^{(k+1)} \\ &> [I_n - \mathrm{diag}(Pv^{(k+1)})]u^{(k+1)}. \end{aligned}$$

Since $I_n - \mathrm{diag}(Pv^{(k+1)}) > I_n - \mathrm{diag}(Pv^*) > 0$, using Lemma 1 and the fact that $v^{(k+1)} < v^*$, we get

$$0 \le u^{(k+1)} < u^{(k+2)}.$$

Also,

$$\begin{aligned} [I_n - \mathrm{diag}(Pv^{(k+1)})]u^{(k+2)} = e &= [I_n - \mathrm{diag}(Pv^*)]u^* \\ &< [I_n - \mathrm{diag}(Pv^{(k+1)})]u^*. \end{aligned}$$

Therefore, $u^{(k+2)} < u^*$ holds, and consequently (i) holds for $(k+1)$.
In conclusion, (i) is proved by induction.

The proof of (ii) can be done via (i) which provides the existence of two positive vectors $(\hat{u}^*, \hat{v}^*)$ where $0 < \hat{u}^* < u^*$ and $0 < \hat{v}^* < v^*$ and satisfying

$$\lim_{k\to\infty} u^{(k)} = \hat{u}^* \qquad \text{and} \qquad \lim_{k\to\infty} v^{(k)} = \hat{v}^*,$$

and

$$\begin{cases} \hat{u}^* = \phi_P(\hat{v}^*), \\ \hat{v}^* = \phi_Q(\hat{u}^*), \end{cases} \tag{3.34}$$

i.e., $(\hat{u}^*, \hat{v}^*)$ is a positive solution of (3.31). Due to the minimal property of (u^*, v^*) and the comparative property of $(\hat{u}^*, \hat{v}^*)$ with (u^*, v^*), it must hold that $\hat{u}^* = u^*$ and $\hat{v}^* = v^*$. □

3.5.6 Computation of the Jacobian matrix

Now, we consider the calculation of the Jacobian matrix of the modified iteration (3.32).

Define

$$\begin{array}{rccl} \phi_{K,L}: & \mathbb{R}^n & \longrightarrow & \mathbb{R}^n, \\ & t & \longrightarrow & \phi_K(\phi_L(t)). \end{array}$$

and

$$\begin{array}{rccl} \Phi: & \mathbb{R}^n \times \mathbb{R}^n & \longrightarrow & \mathbb{R}^n \times \mathbb{R}^n, \\ & (u,v) & \longrightarrow & (\phi_{P,Q}(u), \phi_{Q,P}(v)) \end{array}$$

Then the iterative scheme (3.32) can be written as

$$\begin{bmatrix} u^{(k+1)} \\ v^{(k+1)} \end{bmatrix} := \begin{bmatrix} \phi_{P,Q}(u^{(k)}) \\ \phi_{Q,P}(v^{(k)}) \end{bmatrix}. \tag{3.35}$$

The Jacobian matrix is given by

$$\begin{aligned} J(\Phi,(u,v)) &= \begin{bmatrix} J(\phi_{P,Q},(u,v)) \\ J(\phi_{Q,P},(u,v)) \end{bmatrix} \\ &= \begin{bmatrix} J(\phi_P,\phi_Q(u))J(\phi_Q,u) & 0 \\ 0 & J(\phi_Q,\phi_P(v))J(\phi_P,v) \end{bmatrix}. \end{aligned}$$

At the solution (u^*,v^*), the Jacobian is of the form

$$J(\Phi,(u^*,v^*)) = \begin{bmatrix} J(\phi_P,v^*)J(\phi_Q,u^*) & 0 \\ 0 & J(\phi_Q,u^*)J(\phi_P,v^*) \end{bmatrix}, \tag{3.36}$$

because $\phi_Q(u^*) = v^*$ and $\phi_P(v^*) = u^*$. We have,

$$(I_n - \mathrm{diag}(Pv^*))^{-1}e = \begin{bmatrix} \frac{1}{1-\sum_{j=1}^n P_{1j}v_j^*} \\ \frac{1}{1-\sum_{j=1}^n P_{2j}v_j^*} \\ \vdots \\ \frac{1}{1-\sum_{j=1}^n P_{nj}v_j^*} \end{bmatrix}.$$

Therefore,

$$J(\phi_P, v^*) = \begin{bmatrix} \frac{P_{11}}{(1-\sum_{j=1}^n P_{1j}v_j^*)^2} & \cdots & \frac{P_{1n}}{(1-\sum_{j=1}^n P_{1j}v_j^*)^2} \\ \vdots & \ddots & \vdots \\ \frac{P_{n1}}{(1-\sum_{j=1}^n P_{nj}v_j^*)^2} & \cdots & \frac{P_{nn}}{(1-\sum_{j=1}^n P_{nj}v_j^*)^2} \end{bmatrix} = \{(I_n - \mathrm{diag}(Pv^*))^{-1}\}^2 P.$$

Define

$$\begin{aligned} K: \quad \mathbb{R}^{n\times n} \times \mathbb{R}^n &\longrightarrow \mathbb{R}^{n\times n}, \\ (Y,t) &\longrightarrow \{(I_n - \mathrm{diag}(Yt))^{-1}\}^2 Y. \end{aligned}$$

Then

$$J(\phi_P, v^*) = K(P, v^*). \tag{3.37}$$

Similarly, we have

$$J(\phi_Q, u^*) = K(Q, u^*). \tag{3.38}$$

Subtituting (3.37) and (3.38) into (3.36), the Jacobian matrix at the solution becomes

$$J(\Phi, (u^*, v^*)) = \begin{bmatrix} K(P,v^*)K(Q,u^*) & 0 \\ 0 & K(Q,u^*)K(P,v^*) \end{bmatrix}. \tag{3.39}$$

Conclusion

We were concerned with a special kind of algebraic Riccati equation which arises in transport theory.

We talked about the history of this equations and the existence of nonnegative solutions where only the minimal positive solution is of a physical interest. The matrix form of these equations is derived and an efficient method of getting the minimal positive solution is presented. This method consists of obtaining this solution by computing the minimal positive solution of a vector equation, which is derived by exploiting the special structure of the coefficient matrices of the Riccati equation.

A simple iteration of Lu was developed to compute the minimal positive solution of this vector equation. Bao, Lin, and Wei [3] suggested a modified version of this simple iteration which succeeded to be more efficient. Also, nonlinear splitting iteration methods were discussed and a comparison of all these methods was summarized. However, Newton's method was proposed in [45] to solve the vector equation. But it requires one LU factorization [20], for a matrix of order n, at each iteration. Later, an interesting fast Newton method of Bini et al. [8] was presented. We will compare the results of our approach with this method later in this thesis.

Based on the iteration of Lin which is explained, we constructed a modified version which is more efficient and which allows to do the computations in almost half the cost. The convergence of the modified iteration towards the same minimal positive solution of the vector equation is proved followed by the computation of the Jacobian matrix.

Application of the Reduced Rank Extrapolation Method to NARE

CHAPTER 4

Application of the Reduced Rank Extrapolation Method to NARE

We are interested in applying the restarted reduced rank extrapolation method (RRE) described in Chapter 2 on page 39 to the modified iteration scheme

$$\begin{bmatrix} u^{(k+1)} \\ v^{(k+1)} \end{bmatrix} = \begin{bmatrix} \phi_{P,Q}(u^{(k)}) \\ \phi_{Q,P}(v^{(k)}) \end{bmatrix}. \tag{4.1}$$

to solve the nonsymmetric algebraic matrix Riccati equation (NARE) of Page 54 that arises in transport theory.

4.0.7 Different ways for application

Three approaches for applying the restarted RRE to iteration (4.1) are detailed in Tables 4.1, 4.2, and 4.3. Note that these approaches with their numerical experiments are accepted to be published in ETNA (Electronic Transactions on Numerical Analysis) Journal. We start by the first two tables, then the third table will be discussed later. Table 4.1 applies a restarted RRE method directly to the vector sequence $\{w^{(k)}\}_{k\in\mathbb{N}}$ where $w^{(k)} = [(u^{(k)})^T, (v^{(k)})^T]^T$, while Table 4.2 applies a restarted RRE method to the vector sequences $\{u^{(k)}\}_{k\in\mathbb{N}}$ and $\{v^{(k)}\}_{k\in\mathbb{N}}$ separately. Note that we are working in a space of dimension $2n$. Both approaches accelerate the convergence of the vector sequences but Figure 4.7 shows that the technique of Table 4.1 works better near the critical case. For later numerical experiments, we use the algorithm of Table 4.1.

The restarted RRE(r) **applied to** $\{w^{(k)}\}_k$ **,** r **is fixed**

1. For $k = 0, u^{(0)} = v^{(0)} = 0$, choose an integer r.
   Then, $w^{(0)} = [(u^{(0)})^T, (v^{(0)})^T]^T = 0$.

2. For $k = 1, 2, ...,$
   $y^{(0)} = u^{(k-1)}$; $z^{(0)} = v^{(k-1)}$; $s^{(0)} = (y^{(0)}, z^{(0)})^T$;
   $y^{(j+1)} = \phi_P(z^{(j)})$; $\quad j = 0, \ldots, r-1$.
   $z^{(j+1)} = \phi_Q(y^{(j+1)})$;
   $s^{(j+1)} = (y^{(j+1)}, z^{(j+1)})^T$;
   Compute the approximations $t^{(r-1)}$ by applying the RRE algorithm of Table 2.2 on the vectors $(s^{(0)}, s^{(1)}, \ldots, s^{(r)})$;
   If $t^{(r-1)}$ satisfies accuracy test, stop.
   Else
   $w^{(0)} = (u^{(0)}, v^{(0)})^T = t^{(r-1)}$.
   End

Table 4.1: The restarted RRE(r) applied to $\{w^{(k)}\}_k$, r is fixed.

The restarted RRE(r) **applied to** $\{u^{(k)}\}_k$ **and** $\{v^{(k)}\}_k$ **,** r **is fixed**

1. For $k = 0, u^{(0)} = v^{(0)} = 0$, choose an integer r.

2. For $k = 1, 2, ...,$
   $y^{(0)} = u^{(k-1)}$; $z^{(0)} = v^{(k-1)}$;
   $y^{(j+1)} = \Phi_P(z^{(j)})$, $z^{(j+1)} = \Phi_Q(y^{(j+1)})$, $j = 0, \ldots, r-1$.
   Compute the approximations $t_1^{(r-1)}$ and $t_2^{(r-1)}$ by applying the RRE algorithm of Table 2.2 on the vectors $[(y^{(0)}, z^{(0)})^T, (y^{(1)}, z^{(1)})^T, \ldots, (y^{(r)}, z^{(r)})^T]$;
   If $t_1^{(r-1)}$ and $t_2^{(r-1)}$ satisfy accuracy test, stop.
   Else
   $u^{(0)} = t_1^{(r-1)}$; , $v^{(0)} = t_2^{(r-1)}$.
   End

Table 4.2: The restarted RRE(r) applied to $\{u^{(k)}\}_k$ and $\{v^{(k)}\}_k$, r is fixed.

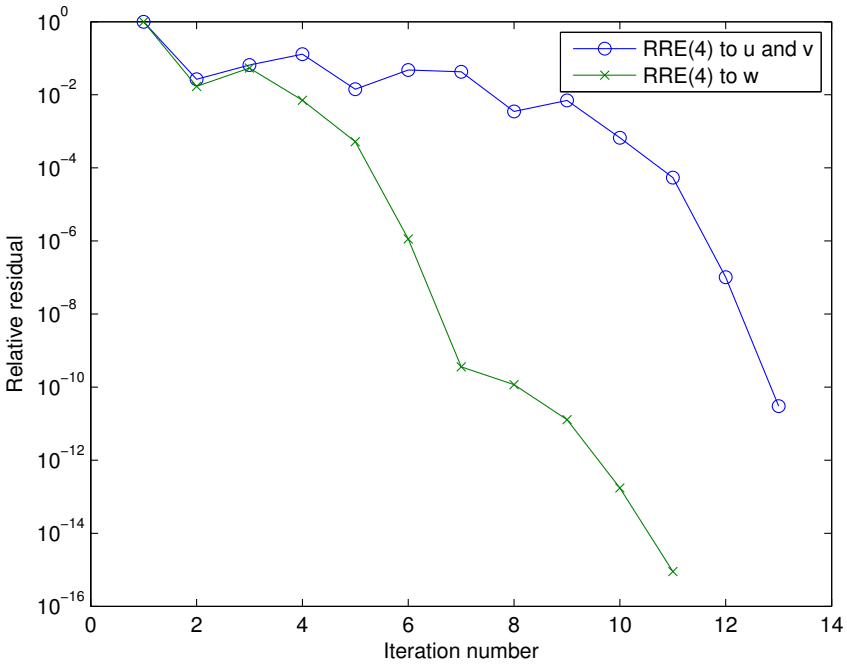

Figure 4.1: $n = 512,\ \alpha = 0.001,\ c = 0.999.$

Now, we propose a third approach for applying the restarted RRE to iteration (4.1). Here, we will apply the restarted RRE to the vector sequence $\{v^{(k)}\}_{k\in\mathbb{N}}$ only

$$v^{(k+1)} = \Phi_{Q,P}(v^{(k)}), \quad \text{with } v^{(0)} = 0. \tag{4.2}$$

Then, when we get the new approximation $t^{(k)}$ of $v^{(k)}$, we compute $u^{(k)}$ by

$$u^{(k)} = \Phi_P(t^{(k)})$$

Note that we are working in a space of dimension n and not $2n$ as in Tables 4.1 and 4.2. We summarize the algorithm in details in Table 4.3 .

The restarted RRE(r) **applied to** $\{v^{(k)}\}_k$ **,** $v^{(k+1)} = \Phi_{Q,P}(v^{(k)})$ **,** r **is fixed**

1. For $k = 0, u^{(0)} = v^{(0)} = 0$, choose an integer r.

2. For $k = 1, 2, ...,$
   $y^{(0)} = v^{(k-1)}$;
   $y^{(j+1)} = \Phi_{Q,P}(v^{(j)}),\ j = 0, \ldots, r-1.$
   Compute the approximations $t^{(r-1)}$ by applying the RRE algorithm of Table 2.2 on the vectors $(y^{(0)}, y^{(1)}, \ldots, y^{(r)})$;
   If $t^{(r-1)}$ satisfies accuracy test, stop.
   Else
   $v^{(0)} = t^{(r-1)}$.
   End

3. $u^{(k)} = \Phi_P v^{(k)}$

Table 4.3: The restarted RRE(r) applied to $\{v^{(k)}\}_k$, $v^{(k+1)} = \Phi_{Q,P}(v^{(k)})$, r is fixed.

4.0.8 Comparison between the three proposed approaches

Here, the example of Page 79 is used. We compare the three approaches of applying the restarted RRE method of Tables 4.1, 4.2, and 4.3 for $n = 512$ and for different (α, c) with choosing the number of restarts r which gives the best results. Here, a small r (3 or 4) is a good choice.

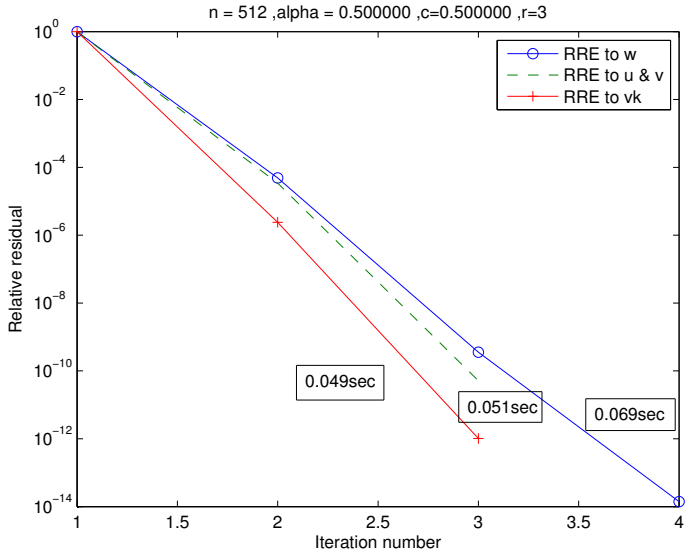

Figure 4.2: Comparison between the three proposed approaches, $(\alpha, c) = (0.5, 0.5)$, $r = 3$.

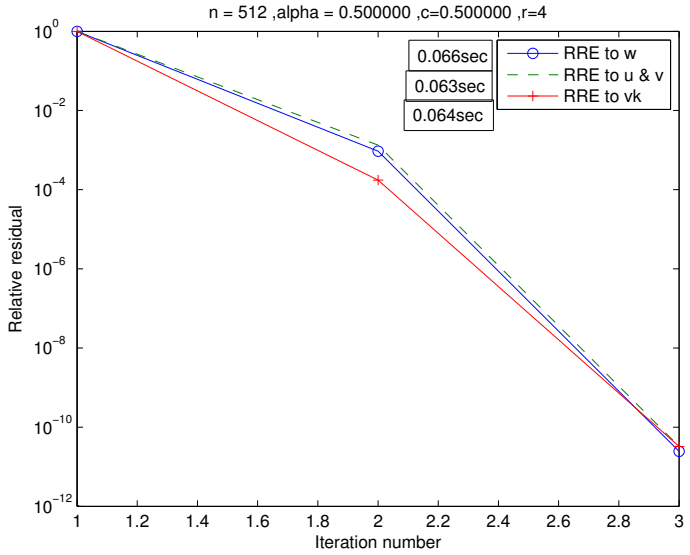

Figure 4.3: Comparison between the three proposed approaches, $(\alpha, c) = (0.5, 0.5)$, $r = 4$.

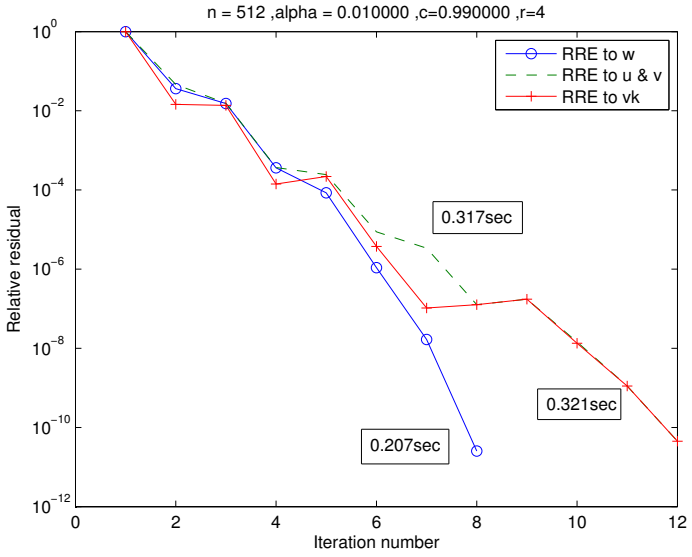

Figure 4.4: Comparison between the three proposed approaches, $(\alpha, c) = (0.01, 0.99)$, $r = 4$.

4.0.9 The choice of r

The RRE algorithm should be restarted every r iterations, for some integer $r > 1$, to avoid the increase in computational work and storage as k increases.

We always have $r \leq N$, where N is the dimension of the system, and sometimes r is much smaller than N. In practice, r is not known but all of the extrapolation methods may be applied with the number of "dominant" eigen values of the sequence generator.

We consider the sequence $\{w^{(k)}\}_{k\in\mathbb{N}}$ of equation (3.32) on Page 63.
If $w^* = \Phi(w^*)$ is a fixed point and $J(\Phi, w^*)$ is the Jacobian matrix of Φ at w^*, then

$$w^{(k+1)} - w^* = J(\Phi, w^*)(w^{(k)} - w^*) + O(\|(w^{(k)} - w^*)\|^2).$$

It suffices to examine the eigenvalues of the Jacobian matrix $J(\Phi, w^*)$. Going back to the iteration (3.32) and observing the eigenvalues of the Jacobian matrix (3.39) on Page 66 at the solution, it can be seen that these eigenvalues range between zero and one. Most of these eigenvalues are close or equal to zero, except for a few which are close to one. Therefore, choosing a small integer r is enough. Figure 4.5 shows the distribution of the spectrum of the Jacobian matrix at the solution for $n = 512$, $(\alpha, c) = (0.001, 0.999)$ with spectral radius $\rho(J(\Phi, (u^*, v^*))) \approx 0.86$.

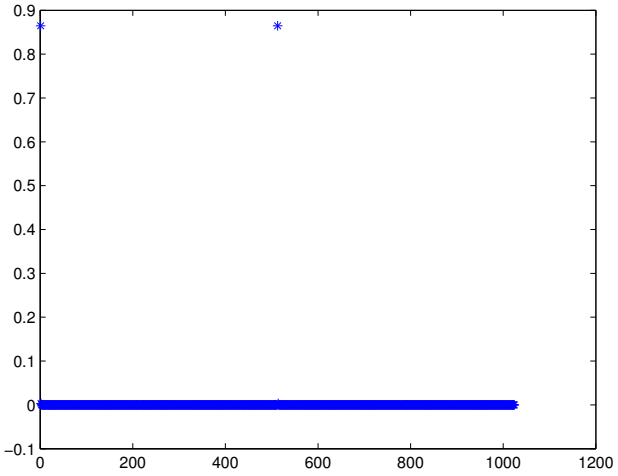

Figure 4.5: Distribution of the spectrum of Jacobian matrix at the solution.

Now, Figures 4.6,4.7, and 4.8 show the convergence of the extrapolated iteration using the algorithm of Table 4.3 for different choices of r. It can be seen that for values of α and c away from 0 and 1 respectively, $r = 3$ is the best choice. While near the critical case, where the convergence becomes really slow, the best choice of r was seen to be 4.

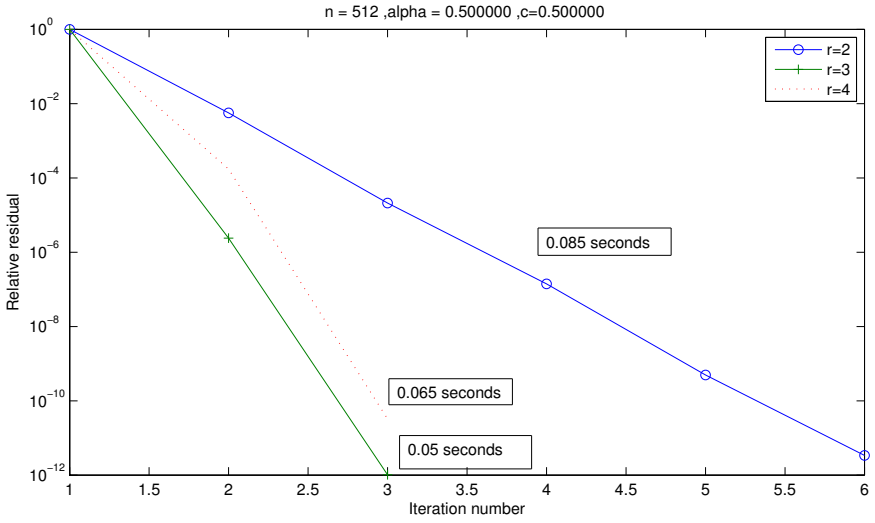

Figure 4.6: Restarted RRE to $\{v^{(k)}\}$ for different choices of r, $(\alpha, c) = (0.5, 0.5)$.

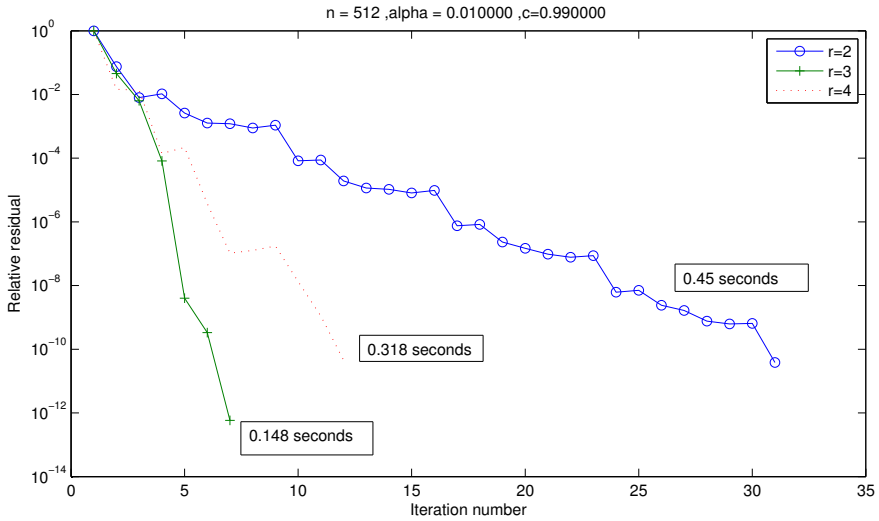

Figure 4.7: Restarted RRE to $\{v^{(k)}\}$ for different choices of r, $(\alpha, c) = (0.01, 0.99)$.

4.1 Numerical Experiments and Comparisons

A numerical example is presented in this section to illustrate the performance of the new approach for solving the vector equation (3.28).

4.1.1 Example

We consider a special kind of Riccati equation (3.7)-(3.8). The constants c_i and ω_i are given by a numerical quadrature formula on the interval $[0,1]$, which is obtained by dividing $[0,1]$ into $n/4$ subintervals of equal length and applying a composite Gauss-Legendre quadrature with 4 nodes in each subinterval.

Computations are performed for different choices of the parameters (α, c) and for different values of n using MATLAB 7.4 (R2007a) on Linux with 2.5 Ghz and with about 15 significant decimal digits. The stopping criterion is given by

$$ERR = \frac{\| w^{(k+1)} - w^{(k)} \|}{\| w^{(k+1)} \|} \leq tol,$$

for $tol = 10^{-10}$.

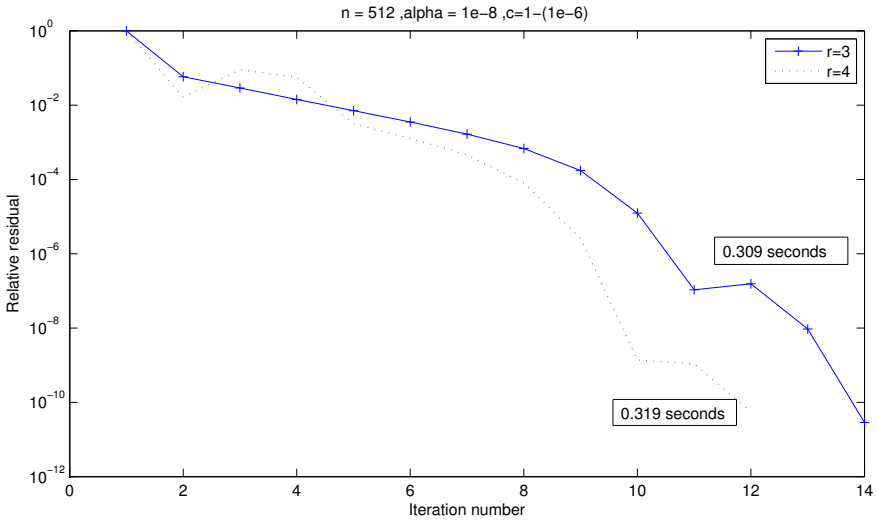

Figure 4.8: Restarted RRE to $\{v^{(k)}\}$ for different choices of r, $(\alpha, c) = (0.00000001, 0.999999)$.

4.1.2 Comparisons and numerical results

These comparisons show the effectiveness of applying vector extrapolation methods, and in particular polynomial extrapolation methods, as convergence accelerators methods on nonlinear systems of equations. The development of restarted methods allows, furthermore, to limit the cost of computations and storage. Then, restarted (or cyclic) algorithms of RRE, MPE, and MMPE are implemented.

Stability of RRE for big r

Table (4.4) and (4.5) compares the three vector extrapolation methods RRE,MPE, and MMPE for two values of r, where r denotes the number of restarts to be done while applying the extrapolation algorithms in the cycling mode (r refer to the size of extrapolation). It is noticed that for $r = 4$, the three methods are comparable and give close results. While for a bigger r ($r = 10$), the RRE method is always the best among them and remains the most stable.

r=4	RRE	MPE	MMPE
CPU time (in seconds)	3.8	3.8	3.9
Residual norm	$3.5.10^{-11}$	2.10^{-11}	$3.9.10^{-11}$

Table 4.4: Comparison of RRE,MPE, and MMPE for $r = 4$ and $n = 2048$.

r=10	RRE	MPE	MMPE
CPU time (in seconds)	6.59	7.57	10.36
Residual norm	$6.87.10^{-12}$	9.0^{-12}	$1.0.10^{-11}$

Table 4.5: Comparison of RRE,MPE, and MMPE for $r = 10$ and $n = 2048$.

Comparison of iterative methods

Table 4.6 compares the iterative method proposed by Lin (3.18), its modified version (3.20), and the application of RRE to $\{w^{(k)}\}$ of Table 4.1. Denote by ILin the iterative method proposed by Lin [44] with the choice of T_k given in (3.29), by MILin its modified version, and by RRE the modified version with the application of restarted reduced rank extrapolation every 4 iterations of Table 4.1. Table 4.6 shows how the three methods converge to the minimal positive solution of (3.31) for several α and c values. The RRE method is seen to outperform ILin and MILin. Figure 4.9 shows the performance of RRE in comparison with ILin and MILin for $n = 256$ and $(\alpha, c) = (0.001, 0.999)$.

α	c	Method	Iteration steps	Residual norm	CPU time
10^{-8}	$1-10^{-6}$	ILin	4732	9.98e-11	5.47
		MILin	2517	9.98e-11	2.87
		RRE	20	5.03e-14	0.11
10^{-5}	$1-10^{-5}$	ILin	1813	9.97e-11	2.08
		MILin	955	9.93e-11	1.08
		RRE	7	3.83e-14	0.05
0.0001	0.9999	ILin	674	9.98e-11	0.77
		MILin	353	9.88e-11	0.4
		RRE	7	9.8e-16	0.04
0.001	0.999	ILin	246	9.7e-11	0.3
		MILin	129	9.04e-11	0.14
		RRE	9	2.99e-14	0.05
0.5	0.5	ILin	12	4.01e-11	0.01
		MILin	7	3.32e-11	0.008
		RRE	3	9.72e-17	0.02

Table 4.6: Numerical results for n=256 with different (α,c).

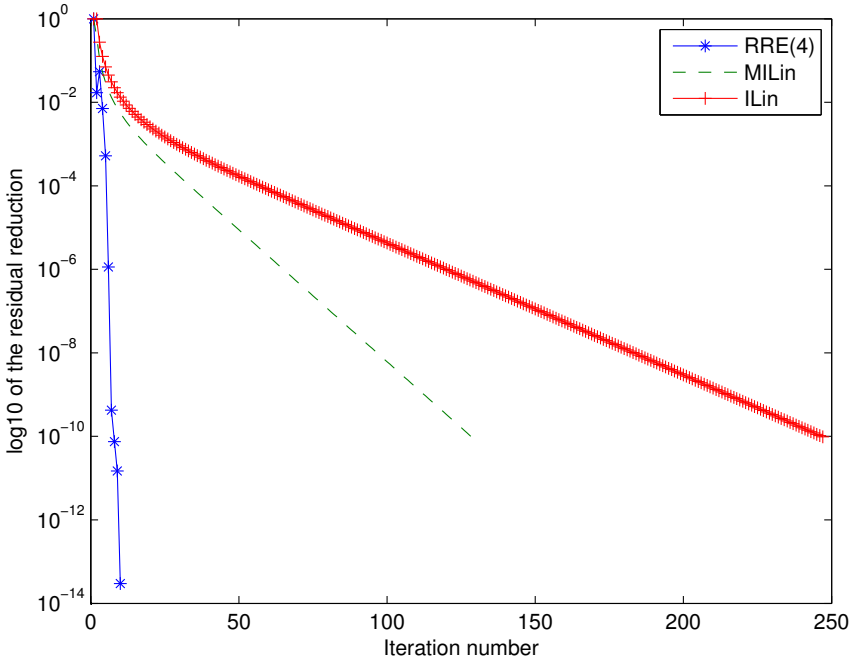

Figure 4.9: $n = 256,\ r = 4,\ (\alpha, c) = (0.001, 0.999)$.

Comparison with the fast Newton method

We now compare RRE of Table 4.1 and the fast Newton method proposed in [8]. Computations of this table have been implemented in Fortran 90 on Linux with 2.5 Ghz and with about 15 significant decimal digits and for $tol = 10^{-10}$. The code Fortran implemented in [8] is used. Denote by LuF the fast Newton method, which is based on a fast LU algorithm that reduces the cost of Newton's algorithm proposed by Lu [45] from $O(n^3)$ to $O(n^2)$; see Section 3.5.3. Table 4.1.2 compares the restarted RRE method with LuF in terms of CPU time (in seconds) for different n and for $(\alpha,\ c) = (10^{-8}, 1-10^{-6})$. It can be seen that RRE is faster than LuF also when the convergence is slow for (α, c) close to $(0,1)$ and for large n.

Remark 4.1.1. *The Fortran code of LuF used a LAPACK function and note that if we optimize our codes with LAPACK, we can get much better and faster results. This code optimization is to be done in the future.*

	$\alpha = 10^{-8},\ c = 1-10^{-6}$	
n	**LuF**	**RRE(4)**
512	0.35	0.24
1024	1.37	0.96
2048	6.6	3.8

Table 4.7: Comparison in terms of CPU time in seconds for different n.

Extrapolation even for large size systems

The RRE, MPE, and MMPE are also applicable to large size systems of equations without any problem of storage. On the other hand, the fast Newton method proposed in [8] faces storage problems when dealing with large systems as it will be working with large size matrices. Below, in Table 4.8, we present some numerical results obtained for RRE, MPE, and MMPE for n large. Note that these computations were also implemented in Fortran 90 on Linux with 2.5 Ghz. Here, the three extrapolation methods were applied in a cyclic mode of the algorithm of Table 4.1,i.e. to $\{w^{(k)}\}$, with similar applications for MPE and MMPE with the example of Page 79 and with $r = 4$, $(\alpha, c) = (0.5, 0.5)$. The size of the system in Table 4.8 is $2n$.

Figure 4.1.2 shows a comparison between the restarted RRE, applied to $\{v^{(k)}\}$ of Table 4.3 for $r = 4$, with the fast Newton method LuF.

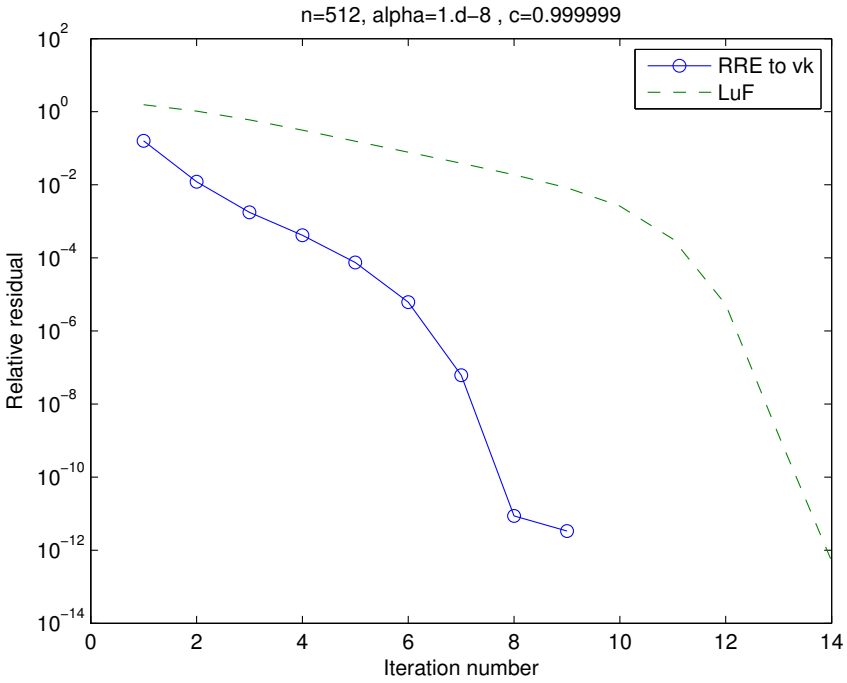

Figure 4.10: $n = 512,\ \alpha = 1.d-8,\ c = 0.999999,\ r = 4.$

n	Method	Residual norm	CPU time in seconds	number of cycles
8000	RRE	1.48d-14	28.04	3
	MPE	1.43d-14	28.20	3
	MMPE	2.26d-14	28.10	3
16000	RRE	2.68d-14	112.49	3
	MPE	2.69d-14	113.27	3
	MMPE	2.55d-14	112.22	3

Table 4.8: The behavior of the polynomial methods on $\{w^{(k)}\}$ for large n, $r = 4$.

Table 4.9 also shows the application of RRE, MPE, and MMPE to large systems of equations but with applying the algorithm of Table 4.3, i.e. to $\{v^{(k)}\}$,

with similar applications for MPE and MMPE, in the cycling mode. The size of the system in Table 4.9 is n, $(\alpha, c) = (0.5, 0.5)$.

n	Method	Residual norm	CPU time in seconds	number of cycles
8000	RRE	3.83d-15	15.33	2
	MPE	3.85d-15	15.01	2
	MMPE	9.32d-15	14.99	2
16000	RRE	5.77d-15	60.91	2
	MPE	5.77d-15	60.91	2
	MMPE	1.16d-14	60.17	2
40000	RRE	9.11d-15	376.28	2
	MPE	9.13d-15	375.36	2
	MMPE	2.08d-14	376.91	2

Table 4.9: The behavior of the polynomial methods on $\{v^{(k)}\}$ for large n, $r = 3$.

Conclusion

We were concerned with the application of extrapolation methods, and in particular, the polynomial vector extrapolation methods to accelerate the convergence of iterative methods.

We proposed different ways of applications of these extrapolation methods where their algorithms were used in a cycling mode to reduce the work requirement which grows quadratically with the number of iteration steps k and the storage requirement which grows linearly with k. The choice of the size of extrapolation used in the restarted algorithms was discussed based on a study of the spectral radius of the Jacobian matrix at the solution.

Numerical experiments and comparisons were made to assure the benefit of using these types of extrapolation methods as convergence accelerators to obtain the minimal positive solution of the required Riccati equation.

The critical case

CHAPTER 5

The critical case

5.1 The Shift technique

In this section, we present the shift technique which is a solution of the problem of convergence in the critical case when $(\alpha, c) = (0, 1)$. The advantage that we get with this technique is twofold: on one hand we can accelerate the speed of the vector iteration by switching from the linear to the quadratic convergence; on the other hand we may guarantee the full machine accuracy in the solution which otherwise would be $O(\sqrt{\epsilon})$.

The shift technique was originally introduced by He, Meini and Rhee for a quadratic matrix equation arising in the numerical solution of Markov chains modeling quasi-birth-and-death (QBD) processes [29].

In the critical where the Jacobian at the solution is singular, the convergence of the iteration scheme (3.31) on Page 63, with the application of the restarted RRE method, turns to linear;

In this case, it is possible to get rid of the singularity of the Jacobian. The idea is to apply the shift technique originally introduced in [29] and used in the framework of Riccati equations by Guo, Iannazzo, and Meini in [25] and by Guo [22]. With this technique, we replace the original Riccati equation with a new one having the same minimal solution as the original equation (1.1) but where the singularity of the Jacobian is removed and the convergence returns quadratic.

It is noticed that the approximation to the minimal positive solution of the iteration (3.32) on Page 63, which leads to the computation of the minimal positive solution of the nonsymmetric algebraic Riccati equation (NARE) (5.1), after the use of the shift technique is more accurate than the one obtained without shift. This will be seen later in the comparisons.

5.1.1 Preliminaries

For any matrices $A, B \in \mathbb{R}^{m\times n}$, we write $A \geq B (A > B)$ if $a_{ij} \geq b_{ij} (a_{ij} > b_{ij})$ for all i, j. A real square matrix A is called a Z-matrix if all its off-diagonal elements are nonpositive. Any Z-matrix A can be written as $sI - B$ with $B \geq 0$. A Z-matrix A is called an M-matrix if $s \geq \rho(B)$, where $\rho(.)$ is the spectral radius; it

is called a singular M-matrix if $s = \rho(B)$ and a nonsingular M-matrix if $s > \rho(B)$.

Lemma 2. *For a Z-matrix A it holds that*

1. *A is an M-matrix if and only if there exists a vector $v > 0$ such that $Av \geq 0$ or a vector $w > 0$ such that $w^T A \geq 0$;*
2. *If A is nonsingular, then A is an M-matrix if and only if $A^{-1} \geq 0$.*

5.1.2 Idea of the Shift

We consider the same nonsymmetric algebraic Riccati equation (NARE) arising in transport theory

$$XCX - XD - AX + B = 0 \tag{5.1}$$

where A, B, C, D are real $n \times n$ matrices, associated to an M-matrix M

$$M = \begin{bmatrix} D & -C \\ -B & A \end{bmatrix}. \tag{5.2}$$

This thesis concerns algebraic Riccati equations associated with nonsingular or singular irreducible M-matrices. The case in which M is singular and reducible is of minor interest.

A useful technique frequently encountered in the theory of matrix equations consists in relating the solutions to some invariant subspaces of a matrix polynomial.

In particular, the solutions of the NARE (5.1) can be put in correspondence with certain n-dimensional invariant subspaces of the matrix

$$H = \begin{bmatrix} D & -C \\ B & -A \end{bmatrix}. \tag{5.3}$$

obtained by premultiplying the M-matrix M defined in (5.2) by the matrix $K = \mathrm{diag}(I_n, -I_n)$. In fact, the matrix H has a double zero eigenvalue corresponding to a 2×2 Jordan block (see [25]).

More precisely, a matrix $X \in \mathbb{R}^{n \times n}$ is a solution of (5.1) if and only if the columns of $\begin{bmatrix} I_n \\ X \end{bmatrix}$ span an invariant subspace of H. In particular, it holds that

$$H \begin{bmatrix} I_n \\ X \end{bmatrix} = \begin{bmatrix} I_n \\ X \end{bmatrix} (D - CX), \tag{5.4}$$

and the eigenvalues of $(D - CX)$ are a subset of the eigenvalues of H.

We say that the NARE (5.1) is associated with the matrix H of (5.3) or that H is the linearizing matrix of the NARE.

In the case of a NARE associated with an M-matrix M of (5.2), it can be proved that the eigenvalues of H can be ordered by non-increasing real part such that

$$0 \leq Re(\lambda_n) \leq Re(\lambda_{n-1}) \ldots \leq Re(\lambda_1) \tag{5.5}$$

In the critical case, the minimal nonnegative solution of (5.1) is ill-conditioned [24] and the convergence of most iterations degrades from quadratic to linear. This is the case when the Jacobian matrix at the solution turns to be singular and this does not guarantee the quadratic convergence of the iterative method used. Most of these problems can be overcome by using the so-called shift technique [25, 29].

Lemma 3. *[25] Let Y be a singular matrix with $Y\nu = 0$ for some nonzero vector ν. If p is a vector so that $p^T\nu = 1$, then for any scalar p, the eigenvalues of the matrix*

$$\bar{Y} = Y + \eta\nu p^T$$

consist of those of Y, except that one zero eigenvalue of Y is replaced by η.

Proof. [25] We may easily verify that $(\bar{Y} - \tau I)\tau I = (Y - \tau I)(\tau I - \eta\nu p^T)$ for any complex number τ. Taking determinants, then we have

$$\tau^n \det(\bar{Y} - \tau I) = \tau^{n-1}(\tau - \eta)\det(Y - \tau I)$$

from which the proof follows. □

The shift technique consists in making a special rank-one correction of H, obtaining a new Riccati equation with the same minimal solution. The new equation has better conditioning, and the convergence of iterations is quadratic again. This rank-one correction of H gives

$$\bar{H} = H + \eta\nu p^T,$$

where $\eta > 0$, ν is a right eigenvector of H corresponding to the zero eigenvalue, and p is an arbitrary vector such that $p^T\nu = 1$. In particular, we write $p^T = [e^T, q^T]$. Under the assumptions (3.8) and (3.9) of Page 54, a right eigenvector of H corresponding to zero is $\nu = [\nu_1^T, \nu_1^T]^T$, where $\nu_1 = \Gamma^{-1}q$ and $\nu_2 = \Delta^{-1}e$.

The rank-one correction is

$$\bar{H} = H + \eta \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} p^T,$$

where $0 < \eta \leq \gamma_1$, $p^T = [e^T \;\; q^T]$, and $p^T v = 1$.

Guo, Iannazo, and Meini [25] proved that $\bar{H}$ has a simple zero eigenvalue. Then, two matrices $\bar{H}$ and $\bar{M}$ are denoted by

$$\bar{H} = \begin{bmatrix} \bar{D} & -\bar{C} \\ \bar{B} & -\bar{A} \end{bmatrix} \quad \text{and} \quad \bar{M} = \begin{bmatrix} \bar{D} & -\bar{C} \\ -\bar{B} & \bar{A} \end{bmatrix}. \tag{5.6}$$

and defines the new Riccati equation

$$X\bar{C}X - X\bar{D} - \bar{A}X + B = 0, \tag{5.7}$$

with

$$\bar{A} = A - \eta v_2 q^T, \quad \bar{B} = B + \eta v_2 e^T,$$
$$\bar{C} = C - \eta v_1 q^T, \quad \bar{D} = D + \eta v_1 e^T.$$

Also in [25], it is proved that the minimal positive solution of (5.1) is the minimal positive solution of (5.7).

It follows from the specific structure of $\bar{M}$ given in (5.6) that the matrix $\bar{M}$ is irreducible. The nice feature of this rank-one modification is that one zero eigenvalue of $\bar{H}$ will be replaced by the scalar $\eta > 0$. This can be seen by directly applying the following useful lemma shown in [25].

Since H is a singular matrix with $Hv = 0$, using Lemma (3) we conclude that the eigenvalues of H and $\bar{H}$ are the same except that one zero eigenvalue is shifted to η.

Now, it remains to show that the iterative scheme (3.31) can be still possible to be implemented for the new Riccati equation. For $p^T = [e^T \;\; q^T]$, the shifted matrix $\bar{H}$ stays a diagonal plus rank-one matrix; so is $\bar{M} = K\bar{H}$. Then, it remains to prove that $\bar{M}$ is an M-matrix.

We have

$$\begin{aligned} \bar{M} &= K(H + \eta v p^T) \\ &= M + \begin{bmatrix} \eta v_1 \\ -\eta v_2 \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix} \\ &= \begin{bmatrix} \Gamma & 0 \\ 0 & \Delta \end{bmatrix} - \begin{bmatrix} q - \eta v_1 \\ e + \eta v_2 \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix} \end{aligned}$$

After the choice of η as $0 < \eta \leq \gamma_1$ and $q \geq 0$, then

$$q - \eta v_1 = (I_n - \eta \Gamma^{-1}) q \geq 0$$

thus, $\bar{M}$ is a Z-matrix. Applying the Perron-Frobenius theorem to $\rho I - M$, there exist a vector $u > 0$ such that $u^T M = 0$. In the critical case, we have $u^T K v = 0$.

Observe that $u^T K$ is a left eigenvector of $H = KM$ corresponding to the zero eigenvalue and that right an left eigenvectors corresponding to the same eigenvalue in a Jordan block of dimension $n \geq 2$ are orthogonal. Then,

$$u^T \bar{M} = u^T M + \eta u^T K v p^T = 0.$$

Thus, $\bar{M}$ is an M-matrix by part one of Lemma 2.

5.1.3 Comparison: with/without shift

We will give some numerical results and comparisons to show the benefit of using a shift technique in the critical case when $(\alpha, c) = (0, 1)$. Note that the computations here were made using MATLAB 7.4 (R2007a) on Linux with 2.5 Ghz and CPU time is in seconds. We use the same example of Section 4.1.

We remark that the approximation to the minimal positive solution of the iteration (3.32) on Page 63, which leads to the computation of the minimal positive solution of the nonsymmetric algebraic Riccati equation (NARE) (5.1), after the use of the shift technique is more accurate than the one obtained without shift. This can be seen in the following figures and tables.

Figure 5.1 shows the bad convergence of the three approaches: RRE to $\{w^{(k)}\}$ of Table 4.1 of Page 72, RRE to $(\{u^{(k)}\}, \{v^{(k)}\})$ of Table 4.2 of Page 73, and RRE to $\{v^{(k)}\}$ of Table 4.3 of Page 75. This is because of the Jacobian matrix at the solution which becomes singular when $(\alpha, c) = (0, 1)$. Figure 5.2, shows how the shift technique gives better convergence by getting rid of the singularity of the Jacobian matrix at the solution. Even the approximation of the required solution is much accurate.

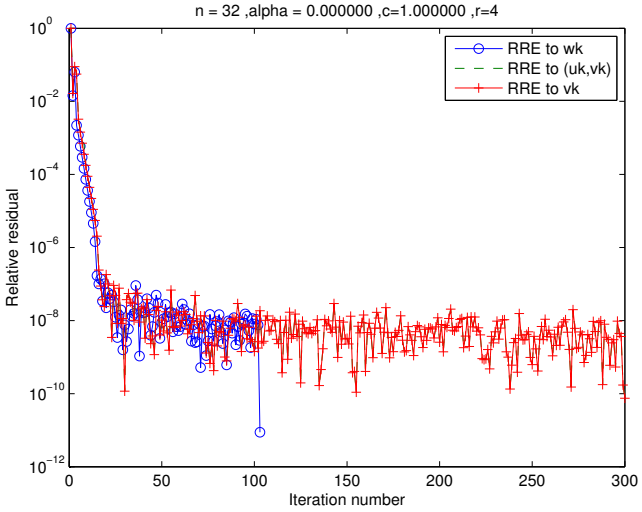

Figure 5.1: Without shift technique.

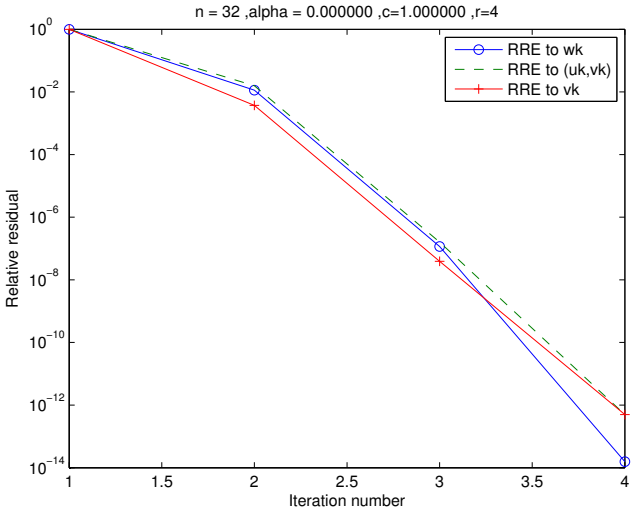

Figure 5.2: With the shift technique.

Tables 5.1, 5.2, and 5.3 present some comparisons between the different approaches to apply the restarted RRE before and after using the shift technique. These comparisons are made in terms in the number of iterations, the residual norm, and the CPU time in seconds.

n	Method	Iteration steps	Residual norm	CPU time
32	RRE to $w^{(k)}$	102	8.80e-12	0.038
	With shift	3	1.55e-14	0.007
64	RRE to $w^{(k)}$	64	5.51e-12	0.03
	With shift	3	1.77e-14	0.007
512	RRE to $w^{(k)}$	90	1.71e-11	2.56
	With shift	3	1.56e-14	0.09
1024	RRE to $w^{(k)}$	204	5.69e-11	19.49
	With shift	3	9.23e-13	0.09

Table 5.1: RRE to $w^{(k)}$ / shift.

n	Method	Iteration steps	Residual norm	CPU time
32	RRE to $(u^{(k)}, v^{(k)})$	299	7.50e-11	0.12
	With shift	3	4.96e-13	0.007
64	RRE to $(u^{(k)}, v^{(k)})$	135	2.49e-11	0.07
	With shift	3	1.27e-12	0.007
512	RRE to $(u^{(k)}, v^{(k)})$	147	7.26e-11	4.09
	With shift	3	9.23e-13	0.09
1024	RRE to $(u^{(k)}, v^{(k)})$	198	9.15e-11	18.61
	With shift	3	9.23e-13	0.09

Table 5.2: RRE to $(u^{(k)}, v^{(k)})$ /shift.

n	Method	Iteration steps	Residual norm	CPU time
32	RRE to $v^{(k)}$	299	7.50e-11	0.12
	With shift	3	4.96e-13	0.007
64	RRE to $v^{(k)}$	135	2.49e-11	0.07
	With shift	3	1.10e-13	0.006
512	RRE to $v^{(k)}$	147	7.26e-11	4.10
	With shift	3	6.87e-13	0.09
1024	RRE to $v^{(k)}$	198	9.15e-11	18.46
	With shift	3	6.87e-13	0.09

Table 5.3: RRE to $v^{(k)}$ / shift.

Comparison with the fast Newton method

A comparison with the fast Newton method (LuF) of Bini et al. [8] is effected. Computations of Table 5.1.3 are implemented in Fortran 90 on Linux with 2.5 Ghz and with about 15 significant decimal digits and for $tol = 10^{-10}$.

	$\alpha = 0,\ c = 1$	
n	**LuF**	**RRE(4) to** $\{v^{(k)}\}$
32	0.35	0.24
512	0.35	0.24
1024	1.37	0.96
2048	6.6	3.8

Table 5.4: Comparison in terms of CPU time in seconds for different n.

5.2 Simplification of the vector iteration

In this section, we shall see what happens in the critical case , i.e. when $\alpha = 0$ and $c = 1$. This case is the most challenging problem while solving NARE

(5.1). We start by a simplification of the vector iteration (3.32) of Page 63.

Looking back at the equation of NARE (3.7) and the equations (3.9) of Page 54, it can be seen that when $\alpha = 0$, we get $\delta_i = \gamma_i = \frac{1}{\omega_i}$, and consequently $\Delta = \Gamma$. We have $T = [t_{i,j}] = [\frac{1}{\delta_i + \gamma_j}]$ and as $\delta_i = \gamma_i$, then $T = [\frac{1}{\delta_i + \delta_j}]$ and T becomes symmetric ($T = T^T$).

On the other hand, using equations (3.14) and (3.15) of Page 56

$$P = [P_{ij}] = [\frac{q_j}{\delta_i + \gamma_j}] = T.\text{diag}(q),$$

and

$$Q = [Q_{ij}] = [\frac{q_j}{\delta_j + \gamma_i}] = T^T.\text{diag}(q) = T.\text{diag}(q),$$

we have, $P = Q$ and consequently $\phi_P = \phi_Q$, where ϕ is defined as in (3.32) on Page 63.

Recall iteration (3.35) of Page 65. For $P = Q$,

$$\begin{bmatrix} u^{(k+1)} \\ v^{(k+1)} \end{bmatrix} = \begin{bmatrix} \phi_{P,P}(u^{(k)}) \\ \phi_{P,P}(v^{(k)}) \end{bmatrix}.$$

Starting with equal initial vectors $u^{(0)} = v^{(0)} = 0$, it can be easily seen by induction that the sequence $\{u^{(k)}\}_{k\in\mathbb{N}}$ and $\{v^{(k)}\}_{k\in\mathbb{N}}$ are equivalent. This implies that it is enough to compute one of the vector sequences, for example $\{u^{(k)}\}_{k\in\mathbb{N}}$

$$u^{(k+1)} = (I_n - \text{diag}(Pu^{(k)})^{-1}e \quad (5.8)$$
$$= \phi_P(u^{(k)}). \quad (5.9)$$

Then, only half of the computational work of (3.32) is needed.

The solution of NARE can be then calculated by

$$X = T \circ (u^*(u^*)^T),$$

where u^* denotes the limit of $u^{(k)}$.

And looking back to Page 66, the Jacobian matrix at the solution can be simplified into the following

$$J(\phi_P, u^*) = [(I_n - \text{diag}(Pu^*)^{-1}]^2 P$$

Conclusion

We were concerned with the study of the critical case where the Jacobian matrix at the solution turns to be singular. In this case, the convergence of the

iterative methods used to compute the minimal positive solution of the vector equation, and consequently, the minimal positive solution of the nonsymmetric algebraic Riccati equation (NARE), turns from being quadratic to linear.

We applied a shift technique to get rid of the singularity of the Jacobian. With this technique, we replaced the original Riccati equation with a new one having the same minimal solution as the original equation but where the singularity of the Jacobian is removed.

Also in the critical case, we succeeded to simplify the modified iteration that we proposed in Section 3.5.5 of Chapter 3. Thus, only half of the computational work was needed.

General Conclusion

In this thesis, we were interested in the study of polynomial vector extrapolation methods and their application as convergence accelerators on iterative methods to solve Algebraic Riccati equations arising in transport theory. However, a modification was done on one of these iterative methods which led to a faster convergence.

In such applications, polynomial extrapolation methods succeeded to accelerate the convergence of these iterative methods, even in the most critical region where the convergence turns to be extremely slow.

The advantage of these methods of extrapolation is that they use a sequence of vectors which is not necessarily convergent, or which converges very slowly to create a new sequence which can admit a quadratic convergence. Furthermore, the development of restarted (or cyclic) methods allowed to limit the cost of computations and storage. Different types of application of these restarted methods were proposed followed by numerical results and comparisons.

An interpretation on the critical case, a challenging problem where the Jacobian matrix at the solution becomes singular and the convergence turns to be linear, was done. This problem was recovered by applying a shift technique to get rid of the singularity of the Jacobian. With this technique, we replaced the original Riccati equation with a new one having the same minimal solution as the original equation but where the singularity of the Jacobian is removed.

Also in the critical case, we succeeded to simplify the modified iteration that we proposed in Section 3.5.5 of Chapter 3. Thus, only half of the computational work was needed.

APPENDIX A

Some Matlab Codes

A.1 Functions used

1. **Function 'vectoruv':**

```
function [zu,zv]=vectoruv(yu,yv)

global P Q e

n=max(size(yu));

 Pyv=P*yv;
 for i=1:n
    zu(i)=1/(1-Pyv(i));
 end
 %for the modified iteration of Lin
 Qzu=Q*zu;
 for i=1:n
     zv(i)=1/(1-Qzu(i));
 end
%for the iteration of Lin
 Qyu=Q*yu;
 for i=1:n
     zv(i)=1/(1-Qyu(i));
 end
```

2. **Function 'parameters':**

```
function [P,T,Q]=parameters(gauss_x,gauss_w)

global n e alpha bet kmax it d g q eta
```

```
%-----------
% Parameters
%-----------

n=512;    % n is multiple of 4
bet=0.999;   % in article: bet is c
alpha=0.001;
kmax=2;

if(mod(n,4)~=0)
error('n must be a multiple of 4');
end

it=0;   %number of iteration
% eps=2.22*10*10^-16; % err=n*eps
wn=zeros(n,1); % vector of nodes
c=zeros(n,1); % vector of weights
e=ones(n,1);
tol=1e-13;

%------------------
% Calculating weights and nodes of the quadrature formula on the
% interval [0,1], the vector q
%------------------

tmp_omega=zeros(n,1);
tmp_ci=zeros(n,1);

gauss_x=[-0.861136311594052%75223946488892809505
            -0.339981043584856%64802665759103244687
            0.339981043584856%64802665759103244687
            0.861136311594052];%75223946488892809505];
gauss_w=[0.347854845137453%57373063949221999 40
            0.652145154862546%42626936050778000 59
            0.652145154862546%42626936050778000 59
            0.347854845137453];%57373063949221999 40];

    for i=0:4:n-1 %points on 0..1
     a=i/n;
     b=(i+4)/n;
        tmp_ci(i+1:i+4)=(b-a)/2*gauss_w;
```

```
        tmp_omega(i+1:i+4)=(b-a)/2*gauss_x+(a+b)/2;
    end

    wn(1:n)=tmp_omega(n:-1:1); %reverses
    c(1:n)=tmp_ci(n:-1:1);

    %---------------------
    % Calculating vect q ,  delta:d and gamma:g
    %---------------------

     d=1./(bet*wn*(1+alpha));
     g=1./(bet*wn*(1-alpha));
     q=0.5*c./wn;

    %---------------------
    % shift!
    %-------------

     eta=0; % no shift
     % eta=1;  % shift technique when alpha=0,bet=1;
     % 0<eta<=g(1)<g(2)<...<g(n)  then 0<eta<=1.0010860507924

if eta~=0 && (alpha==0 || bet==1)
disp('WARNING: shifting with a nonsingular matrix')
end

if(eta~=0)
q=q-eta*q./g;
e=e+eta*e./d;
end

%----------------------
% Calculating the cauchy matrix T then P & Q
%----------------------

P=zeros(n);
Q=zeros(n);
for j=1:n
for i=1:n
T(i,j)=1/(d(i)+g(j));
P(i,j)=q(j)/(d(i)+g(j));   % or P=T*diag(q)
```

```
Q(i,j)=q(j)/(d(j)+g(i));   % or Q=T'*diag(q)
end
end
%----------------------


% A=diag(d)-e*q';

% B=e*e';

% C=q*q';

% D=diag(g)-q*e';

% H=[D -C;B -A];
```

A.2 Main codes

1. **Code of the iterative scheme of Lu :**

```
%----------------------------------
% Computation of the minimal positive solution of NARE
%    (I)     0 = R(X):= XCX - XD - AX + B
% via computing the minimal positive solution of a vector equation
% by the iterative scheme of Lin.
% Solution of (I) : X = To(u1.v1^T)
%   where o denotes the Hadamard product
% Input:
% u0 and v0   : initial starting vectors
% alpha, bet  : Physical parameters (0<=alpha<1,0<bet<=1)
% tol         : stopping criterion,
%               i.e., the iteration is stopped if residual norm<=tol
% Output:
% u1,v1         : the extrapolated vectors of u,v
% it          : iteration steps
% err         : residual norm
% elapsed time
% X           : solution of (I)
%
% Rola EL-MOALLEM, 2013
%----------------------------------

clear all

format long

global n e alpha bet gauss_x gauss_w P Q it d g

[P,T,Q]=parameters(gauss_x,gauss_w);

%--------------
% initial values
%--------------

ER1=[];
u0=zeros(n,1);
v0=zeros(n,1);
```

```
err=1;
%------------------
%starting the loop
%------------------

tic
while err>tol
u0=u1;
v0=v1;

[u1,v1]=vectoruv(u0,v0);     % we change in the function vectoruv

err=max((norm(u1-u0,2)/norm(u1,2)),(norm(v1-v0,2)/norm(v1,2)));
ER1=[ER1,err];
it=it+1;
end
toc

%-----------------

iER1=length(ER1);
plot(1:iER1,log10(abs(ER1)));
xlabel('Iteration number');
ylabel('Relative residual');

%-------------------
% Calculating the solution X=T*(u1*(v1)')
%-------------------

for j=1:n
for i=1:n
X(i,j)=u1(i)*v1(j)/(d(i)+g(j));
end
end

%-------
% Output
%-------

u1
```

```
v1
it
err
cpt
X
%---------------------------------%
% Jacobian matrix at the solution %
%---------------------------------%
KQu=inv(eye(n)-diag(Q*u1))*inv(eye(n)-diag(Q*u1))*Q;
KPv=inv(eye(n)-diag(P*v1))*inv(eye(n)-diag(P*v1))*P;
J1=zeros(n);
J2=KPv;
J3=KQu;
J4=zeros(n);
%
J=[J1 J2;J3 J4];
%
if any(eig(J) >= 1),  disp('WARNING: Jacobian J contains at least one eig value

plot(1:length(eig(J)),real(eig(J)),'*');
title('Eigen values of the Jacobian')

%---------%
% Remark: %
%---------%

 A=diag(d)-e*q;

 B=e*e';

 C=q'*q;

 D=diag(g)-q'*e';
```

2. **Code of the modified iterative scheme of Lu :**

```
%-------------------------------------
% Computation of the minimal positive solution of NARE
```

```
%    (I)     0 = R(X):= XCX - XD - AX + B
% via computing the minimal positive solution of a vector equation
% by the modified iterative scheme of Lin.
%
% Solution of (I) : X = To(u1.v1^T)
%   where o denotes the Hadamard product
%
% Input:
% u0 and v0   : initial starting vectors
% alpha, bet  : Physical parameters (0<=alpha<1,0<bet<=1)
% tol         : stopping criterion,
%               i.e., the iteration is stopped if residual norm<=tol
% Output:
% u1,v1         : the extrapolated vectors of u,v
% it          : iteration steps
% err         : residual norm
% elapsed time
% X           : solution of (I)
%
% Rola EL-MOALLEM, 2013
%-------------------------------------

clear all

format long

global n e alpha bet gauss_x gauss_w P Q it d g

[P,T,Q]=parameters(gauss_x,gauss_w);

%--------------
%initial values
%--------------

ER2=[];
u0=zeros(n,1);
v0=zeros(n,1);

err=1;

%------------------
```

```
% starting the loop
%------------------

tic
while err>n*eps
u0=u1;
v0=v1;

[u1,v1]=vectoruv(u0,v0);    % we change in the function vectoruv

err=max((norm(u1-u0,2)/norm(u1,2)),(norm(v1-v0,2)/norm(v1,2)));
ER2=[ER2,err];
it=it+1;
end
toc

%-----------------

iER2=length(ER2);
plot(1:iER2,log10(abs(ER2)));
xlabel('Iteration number');
ylabel('Relative residual');

%-------------------
% Calculating the solution X=T*(u1*(v1)')
%-------------------

for j=1:n
for i=1:n
X(i,j)=u1(i)*v1(j)/(d(i)+g(j));
end
end

%---------------------------------%
% Jacobian matrix at the solution %
%---------------------------------%
  KPv=inv(eye(n)-diag(P*v1))*inv(eye(n)-diag(P*v1))*P;
  KQu=inv(eye(n)-diag(Q*u1))*inv(eye(n)-diag(Q*u1))*P;
  J1=KPv*KQu;
  J2=KQu*KPv;
J=[J1 zeros(n);zeros(n) J2];
```

```
plot(1:length(eig(J)),real(eig(J)),'*');
title('Eigen values of the Jacobian')
%
if any(eig(J) >= 1),  disp('WARNING: Jacobian J contains at least one ei

%---------%
% Remark: %
%---------%

 A=diag(d)-e*q;

 B=e*e';

 C=q'*q;

 D=diag(g)-q'*e';
```

3. **Code of the modified iterative scheme of Lin + RRE :**

```
%--------------------------------------
% Computation of the minimal positive solution of NARE
%    (I)     0 = R(X):= XCX - XD - AX + B
% via computing the minimal positive solution of a vector equation
% by modified iterative scheme of Lin with the application of reduced ra
% extrapolation RRE to (w^k) to accelerate the convergence.
% Solution of (I) : X = To(s.t^T)
%   where o denotes the Hadamard product
% Input:
% u0 and v0   : initial starting vectors
% alpha, bet  : Physical parameters (0<=alpha<1,0<bet<=1)
% kmax        : number of steps for restarting the RRE cycling process
% tol         : stopping criterion,
%               i.e., the iteration is stopped if residual norm<=tol
% Output:
% s,t         : the extrapolated vectors of u,v
% it          : iteration steps
% err         : residual norm
% elapsed time
```

```
% X           : solution of (I)
%
% Rola EL-MOALLEM, 2013
%--------------------------------------

clear all
global n e alpha bet tol kmax gauss_x gauss_w P Q it d g q wn c tmp_omega tmp_c
format long
%------------------------------
    % Numerical and physical inputs
    %------------------------------
    ER4=[];
    [P,T,Q]=parameters(gauss_x,gauss_w);
    %----------------------------------
    u0=zeros(n,1);
    v0=zeros(n,1);
    %---------------
    w0=[u0;v0];
    yu=u0;
       yv=v0;
       yw=[yu;yv];

ndim=length(w0);
         rw=zeros(kmax,kmax);
         qw=zeros(ndim,kmax);

          gammw=zeros(kmax,1);
          xiw=zeros(kmax,1);
          cw=zeros(kmax,1);

        err=1;
        %---------------------------------------
         %starting the loop
        %-------------------
        tic
            while err>tol
             for k=1:kmax
              % computation of xk+1 from xk, and computation of uk
               [zu,zv]=vectoruv(yu,yv);   %i.e. uk+1=vector(uk)
               zw=[zu;zv];
               yw=zw-yw;
```

```
 % computation of qk from uk bu0 the modified gram-schmidt
  if (k==1)
     rw(1,1)=norm(yw);
     qw(:,1)=(1/rw(1,1)).*yw;
  else
      j=1:k-1;
      rw(j,k)=qw(:,j)'*yw;
      yw=yw-qw(:,j)*rw(j,k);
      rw(k,k)=norm(yw)  ;
     if (k<kmax)
         hpw=1d0/rw(k,k)  ;
          qw(:,k)=hpw.*yw  ;
      end
  end
  yu=zu;
  yv=zv;
  yw=zw;
 end

%end of computation of the vector qk

%-------------------------------------------------------------
%computation of the gamma's for rre %gamma here is d in arti
     %solving two triangular systems: Rk triu and Rk' tril
          %(R_k)'* c=e   with  (R_k)': tril
               for i=1:k
                   ciw=1;

                   if (i>1)
                       j=1:i-1 ;
                       ciw=ciw-(rw(j,i))'*cw(j);  %size(rw(
                   end

                   cw(i)=ciw/rw(i,i);
               end
            %R_k*gamma=c with R_k triu
               for i=k:-1:1
                   ciw=cw(i);
                   if (i<k)
                       j=k:-1:i+1;
                       ciw=ciw-rw(i,j)*gammw(j);
```

```
                                end
                                gammw(i)=ciw/rw(i,i);
                            end
            %end of solving the triangular systems
                    ik=1:k;
                    somw=sum(gammw(ik));
                    gammw(ik)=(1/somw).*gammw(ik);
                    %res=1/sqrt(abs(som));
               %  end of computation of the gamma's for rre
               %-------------------------------------------------------
               %  computation of the approximation s(0,k)
                    xiw(1)=1-gammw(1);
                    xiw(2:k)=xiw(1:k-1)-gammw(2:k);
                    t=[u0;v0];%r=w0=[u0;v0]=[ones(n,1);ones(n,1)];

                    for j=1:k
                        jk=j:k-1;

                        hpw=rw(j,jk)*xiw(jk);
                        t=t+hpw*qw(:,j);
                    end

               %end of computation of the approximation s(0,k)
               err=norm(t-yw,2)/norm(t,2)
                ER4=[ER4,err];
                 yw=t;
                 yu=t(1:n);
                 yv=t(n+1:2*n);

               u0=t(1:n);
               v0=t(n+1:2*n);
                w0=t;
              it=it+1;
end
toc
ER4=[1 ER4];
%---------------------------------------
  iER4=length(ER4);
  plot(1:iER4,log10(abs(ER4)))
xlabel('Iteration number');
ylabel('Relative residual');
```

```
%-------------------------
% X=T.*(s*t');
for j=1:n
    for i=1:n
        X(i,j)=yu(i)*yv(j)/(d(i)+g(j));
    end
end

fid=fopen('RREtow.txt','w+');
fprintf(fid, '%6.4e \n', abs(ER4));
fclose(fid);

%------------------------------------------------------------
%Output
%------
cpt
 it
 err_RREtow=err
 norm(X)
%---------%
% Remark: %
%---------%
% A=diag(d)-e*q;
% B=ones(n,n);
% C=q*q';
% D=diag(g)-q*ones(1,n);

%---------------------------------%
% Jacobian matrix at the solution %
%---------------------------------%
u=yu;
v=yv;
KPu=(inv(eye(n)-diag(P*u))*inv(eye(n)-diag(P*u)))*P;
KQu=(inv(eye(n)-diag(Q*u))*inv(eye(n)-diag(Q*u)))*Q;
KPv=(inv(eye(n)-diag(P*v))*inv(eye(n)-diag(P*v)))*P;
KQv=(inv(eye(n)-diag(Q*v))*inv(eye(n)-diag(Q*v)))*Q;

J1=KPv*KQu;
J2=KQu*KPv;
J=[J1 zeros(n);zeros(n) J2];
if any(eig(J) >= 1),  disp('WARNING: Jacobian J contains at least one ei
```

```
plot(1:length(eig(J)),real(eig(J)),'*');
title('Eigen values of the Jacobian')
```

Bibliography

Bibliography

[1] A.C. Aitken, *On Bernoulli's numerical solution of algebraic equations*, Proc. R. Soc. Edinb., 46, pp. 289–305, 1926.

[2] Z.Z. Bai, Y.H. Gao and L.Z. Lu, *Fast iterative schemes for nonsymmetric algebraic Riccati equations arising from transport theory*, SIAM J. Sci. Comput., 30 (2008), pp. 804–818.

[3] L. Bao, Y. Lin, and Y. Wei, *A modified simple iterative method for nonsymmetric algebraic Riccati equations arising in transport theory*, Appl. Math. Comput., 181 (2006), pp. 1499–1504.

[4] R. Bellman and G. M. Wing, *An Introduction to Invariant Embedding*, John Wiley, New York, 1975.

[5] P. Benner and R. Byers, *An exact line search method for solving generalized continuous-time algebraic Riccati equations*, IEEE Trans. Automat. Control, 43 (1), pp. 101–107, 1998.

[6] D. A. Bini, B. Iannazzo, and B. Meini , *Numerical Solution of Algebraic Riccati Equations*, Fundamentals of Algorithms n. 9, SIAM, Softcover, ISBN 978-1-611972-08-5. , 2012.

[7] D.A. Bini, B. Iannazzo, G. Latouche, and B. Meini, *On the solution of algebraic Riccati equation arising in fluid queues*, Linear Algebra Appl., 413, pp. 474–494, 2006.

[8] D.A. Bini, B. Iannazzo and F. Poloni, *A fast Newton's method for a nonsymmetric algebraic Riccati equation*, SIAM J. Matrix Anal. Appl. 30, pp. 276–290, 2008.

[9] C. Brezinski, *Généralisations de la transformation de Shanks, de la table de Padé et de l'ε-algorithme*, Calcolo 12, pp. 317–360 , 1975.

[10] S. Cabay and L.W. Jackson, *A polynomial extrapolation method for finding limits and antilimits for vector sequences*, SIAM J. Numer. Anal. 13, pp. 734–752, 1976.

[11] F. Coron, *Computation of the asymptotic states for linear half space kinetic problem*, Transport Theory Statist. Phys., 19, pp. 89–114, 1990.

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

[13] J.E. DENNIS and R.B. SCHNABEL, *Numerical methods for unconstrained optimization and nonlinear equations*, T. 16. SIAM, Classics in applied mathematics, 1996.

[14] R.P. Eddy, *Extrapolating to the limit of a vector sequence*, In: P.C.C. Wang, Editor, Information Linkage between Applied Mathematics and Industry, Academic Press, New York, pp. 387–396 , 1979.

[15] W.F. Ford and A. Sidi, *Recursive algorithms for vector extrapolation methods* , Appl. Numer. Math. 4, pp. 477–489 , 1988.

[16] B. Gabutti, *An algorithm for computing generalized Euler's transformations of series*, Computing, 34, pp. 107–116, 1985.

[17] B. Gabutti and J. N. Lyness, *Some generalizations of the Euler-Knopp transformation*, Numer. Math., 48, pp. 199–220, 1986.

[18] B. D. Ganapol, *An investigation of a simple transport model*, Transport Theory Statist. Phys., 21, pp. 1-37, 1992.

[19] I. Gohberg, T. Kailath, and V. Olshevsky, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. Comp., 64 (1995), pp. 1557–1576.

[20] G.H. Golub and F.V.L. Charles, *Matrix computations (3rd ed.)*, Baltimore: Johns Hopkins. pp. 257–258 , 1996.

[21] C.H. Guo, *A note on the minimal nonnegative solution of a nonsymmetric algebraic Riccati equation*, Linear Algebra Appl., 357, pp. 299–302, 2002.

[22] C.H. Guo, *Efficient methods for solving a nonsymmetric algebraic Riccati equation arising in stochastic fluid models*, J. Comput. Appl. Math., 192 (2), pp. 353–373, 2006.

[23] —— , *Nonsymmetric algebraic Riccati equations and Weiner-Hopf factorization for M-matrices*, SIAM J. Matrix Anal. Appl., 23, pp. 225–242, 2001.

[24] C.H. Guo and N. J. HIGHAM, *Iterative Solution of a Nonsymmetric Algebraic Riccati Equation*, SIAM. J. Matrix Anal. Appl., 29 (2), pp. 396–412, 2007.

[25] C.H. Guo, B. Iannazzo, and B. Meini, *On the doubling algorithm for a (shifted) nonsymmetric algebraic Riccati equation* SIAM J. Matrix Anal. Appl., 29 (4), pp. 1083–1100, 2007.

[26] C.H. Guo and A.J. Laub, *On the iterative solution of a class of nonsymmetric algebraic Riccati equations*, SIAM J. Matrix Anal. Appl., 22 (2) (2000), pp. 376–391.

[27] C.H. Guo and W.W. Lin, *Convergence rates of some iterative methods for nonsymmetric algebraic Riccati equations arising in transport theory*, Linear Algebra Appl., 432 (2010), pp. 283–291.

[28] C.H. Guo and W.W. Lin, and S.F. Xu *A structure-preserving doubling algorithm for nonsymmetric algebraic Riccati equation*, Numer. Math. 103 (3), pp. 393–412, 2006.

[29] He C, Meini B, Rhee NH, *A shifted cyclic reduction algorithm for quasi-birth-death problems*, SIAM Journal on Matrix Analysis and Applications, 23 (3), pp. 673–691, 2001/02.

[30] K. Jbilou, *A general projection algorithm for solving systems of linear equations*, Numer. Algorithms 4, pp. 361–377 , 1993.

[31] —— , *On some vector extrapolation methods*, Technical Report, ANO(305), Université de Lille1, France, 1993.

[32] K. Jbilou and H. Sadok, *Analysis of some vector extrapolation methods for linear systems*, Numer. Math. 70, pp. 73–89 , 1995.

[33] —— , *LU implementation of the modified minimal polynomial extrapolation method for solving linear and nonlinear systems*, IMA J. Numer. Anal., 19 (4) (1999), pp. 549–561.

[34] —— , *Some results about vector extrapolation methods and related fixed point iterations*, J. Comput. Appl. Math. 36, pp. 385–398 , 1991.

[35] —— , *Vector extrapolation methods. Applications and numerical comparison*, J. Comput. Appl. Math., 122 (2000), pp. 149–165.

[36] D. C. Joyce, *Survey of extrapolation processes in numerical analysis*, SIAM Rev., 13, pp. 435-490, 1971.

[37] J. Juang , *Existence of algebraic matrix Riccati equations arising in transport theory*, Linear Algebra Appl., 230 (1995), pp. 89–100.

[38] J. Juang , *Global existence and stability of solutions of matrix Riccati equations*, J. Math. Anal. Appl., 258(1), pp. 1–12, 2001.

[39] J. Juang and I-Der Chen, *Iterative solution for a certain class of algebraic matrix Riccati equations arising in transport theory*, Transport Theory Statist. Phys., 22 (1993), pp. 65–80.

[40] J. Juang and W.W. Lin, *Nonsymmetric algebraic Riccati equations and Hamiltonian-like matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 228–243.

[41] J. Juang and P. Nelson, *Global existence, asymptotic and uniqueness for the reflection kernel of the angularly shifted transport equation*, Math. Models Methods Appl. Sci., 5, pp. 239–251, 1995.

[42] D. Kleinman, On an iterative technique for riccati equation computations. IEEE Trans. Automat. Control, 13(1), pp. 114–115, 1968.

[43] P. Lancaster and L. Rodman, *Algebraic Riccati equations*, Oxford Science Publications. The Clarendon Press Oxford University Press, New York, 1995.

[44] Y. Lin, *A class of iterative methods for solving nonsymmetric algebraic Riccati equations arising in transport theory*, Computers and Mathematics with Applications 56, pp. 3046–3051, 2008.

[45] L.Z. Lu , *Newton iterations for a non-symmetric algebraic Riccati equation*, Numer. Linear Algebra Appl., 12 (2005), pp. 191–200.

[46] ——, *Solution form and simple iteration of a nonsymmetric algebraic Riccati equation arising in transport theory*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 679–685.

[47] V. L. Mehrmann, *The autonomous linear quadratic control problem*, volume 163 of Lecture Notes in Control and Information Sciences. Springer-Verlag, Berlin, 1991. Theory and numerical solution.

[48] M. Mesina, *Convergence acceleration for the iterative solution of the equations $X = AX + f$*, Comput. Methods Appl. Mech. Engrg. 10, pp. 165–173 , 1977.

[49] W. Niethammer, *Numerical application of Euler's series transformation and its generalizations*, Numer. Math., 34, pp. 271–283, 1980.

[50] J.M. ORTEGA and W.C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, Harcourt Brace Jovanovich, 1970.

[51] R. Penrose, *A generalized inverse for matrices*, Proceedings of the Cambridge Philosophical Society 51: pp. 406–413, 1955.

[52] R. Powell and S. M. Shah, *Summability Theory and its Applications*, Van Nostrand Reinhold, London, 1972.

[53] B.P. Pugatchev, *Acceleration of the convergence of iterative processes and a method for solving systems of nonlinear equations*, USSR. Comput. Math. Math. Phys., pp. 199–207 , 1978.

[54] V. Ramaswami, *Matrix analytic methods for stochastic fluid flows*, In D. Smith and P. Hey, editors, Teletraffic Engineering in a Competitive World, Proceedings of the 16th International Teletraffic Congress, Elsevier Science B.V., Edimburgh, UK, pages pp. 1019–1030, 1999.

[55] L. C. G. Rogers, *Fluid models in queueing theory and Wiener-Hopf factorization of Markov chains*, Ann. Appl. Probab., 4(2), pp. 390–413, 1994.

[56] Y. Saad, *Iterative methods for sparse linear systems (2nd edition)*, SIAM, PWS, 2003.

[57] —— , *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comput. 37, pp. 105–126 , 1981.

[58] Y. Saad and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 72 , pp. 856–869, 1986.

[59] H. Sadok, *Quasilinear vector extrapolation methods*, Linear Algebra Appl. 190, pp. 71–85 , 1993.

[60] —— , *Méthodes de projection pour les systèmes linéaires et non linéaires* , Habilitation Thesis, Université de Lille 1, France, 1994.

[61] D. Shanks, *Nonlinear transformations of divergent and slowly convergent sequences* , J.Math. Phys, vol. 34 , pp. 1–42, 1955.

[62] A. Sidi, *Convergence acceleration for the iterative solution of the equations* $X = AX + f$, Journal of Computational and Applied Mathematics, Vol. 36 , pp. 305–337 , 1991.

[63] —— , *Convergence and Stability Properties of Minimal Polynomial and Reduced Rank Extrapolation Algorithms.*, SIAM J. Numer. Anal. 23, no. 1, pp. 197–209 , 1986.

[64] —— , *Efficient implementation of minimal polynomial and reduced rank extrapolation methods*, J. Comput. Appl. Math., 36 (3) (1991), pp. 305–337.

[65] —— , *Extrapolation vs. Projection Methods for Linear Systems of Equations.*, J. Comput. Appl. Math. 22, pp. 71–88 , 1988.

[66] —— , *Practical extrapolation methods. Theory and applications*, Cambridge Monographs on Applied and Computational Mathematics, 10. Cambridge University Press, Cambridge, 2003.

[67] A. Sidi and J. Bridger, *Convergence and stability analyses for some vector extrapolation methods in the presence of defective iteration matrices.*, Journal of Computational and Applied Mathematics, Vol. 22 , pp. 35–61 , 1988.

[68] A. Sidi and M.L. Celestina, *Convergence acceleration for vector sequences and applications to computational fluid dynamics*, NASA TM-101327, ICOMP-88-17, (August 1988); also: AIAA Paper 90-0338, AIAA 28th Aerospace Sciences Meeting, Reno, NV.

[69] A. Sidi, W.F. Ford, and D.A. Smith *Acceleration of convergence of vector sequences*, SIAM J. Numer. Anal. 23 (1), pp. 178–196 , 1986.

[70] D.A. Smith, W.F. Ford, and A. Sidi, *Extrapolation methods for vector sequences*, SIAM Rev. 29 (1), pp. 199–233 , 1987.

[71] R.R.Tucker, *The $Delta^2$-process and related topics*, Pac. J. Math., 22, pp. 349–359, 1967.

[72] G. Walz, *A counter integral representation on linear extrapolation methods*, Numer. Math., 55, pp. 477–480, 1989.

[73] D. Williams, *A "potential-theoretic" note on the quadratic Wiener-Hopf equation for Q-matrices* In Seminar on Probability, XVI, volume 920 of Lecture Notes in Math., pp. 91–94. Springer, Berlin, 1982.

[74] J. Wimp, *Acceleration methods*, Encyclopedia of Computer Science and Technology, volume 1, New York, 1975.

[75] —— , *Some transformations of monotone sequenses*, Math. Comput., 26, pp. 251–254, 1972.

[76] —— , *Toeplitz arrays, linear sequence transformations and orthogonal polynomials*, Numer. Math., 23, pp. 1–17, 1974.

[77] P. Wynn, *Acceleration techniques for iterated vector and matrix problems*, Mathematics of Computation 16, pp. 301–322 , 1962.

[78] —— , *On a device for computing the $e_m(S_n)$ transformation*, Mathematical Tables and Aids of Computation 10, pp. 91–96 , 1956.

[79] —— , *Transformations to accelerate the convergence of Fourier series*, Gertrude Blanch Anniversary Volume, pp. 339–379, Wright Patterson Air Forces Base, 1967.

Abstract. In this thesis, we are interested in the study of polynomial extrapolation methods and their application as convergence accelerators on iterative methods to solve Algebraic Riccati equations arising in transport theory . In such applications, polynomial extrapolation methods succeed to accelerate the convergence of these iterative methods, even in the most critical region where the convergence turns to be extremely slow. The advantage of these methods of extrapolation is that they use a sequence of vectors which is not necessarily convergent, or which converges very slowly to create a new sequence which can admit a quadratic convergence. Furthermore, the development of restarted (or cyclic) methods allows to limit the cost of computations and storage. We search for the most efficient iterative methods used to solve such kind of Riccati equations. Convergence problems of these methods are examined and critical regions where the convergence turns to be very slow are located. Then, we apply polynomial extrapolation to these iterative methods to improve the convergence, especially in these regions. An interpretation of the critical case which is the most challenging problem is made. In this case, the Jacobian matrix at the required solution is singular and quadratic convergence turns to linear. This problem can be overcome by applying a suitable shift technique in order to get rid of the singularity. The original equation is transformed into an equivalent Riccati equation where the singularity is removed while the matrix coefficients maintain the same structure as in the original equation. The nice feature of this transformation is that the new equation has the same solution as the original one although the new Jacobian matrix at the solution is nonsingular. Numerical experiments and comparisons which confirm the effectiveness of the new approaches are reported.
Keywords: Polynomial vector extrapolation methods, convergence acceleration, restarted algorithms, iterative methods, vector sequences, nonlinear system of equations, algebraic Riccati equation, transport theory, minimal positive solution, Jacobian matrix, critical case.

Résumé. Nous nous intéressons, dans cette thèse, à l'étude des méthodes d'extrapolation polynomiale ainsi qu'à leurs applications à l'accélération de méthodes itératives pour la résolution d'un cas particulier de l'équation algébrique de Riccati utilisée dans la théorie de transport. Pour ce type d'applications, l'extrapolation polynomiale permet d'accélérer la convergence des méthodes itératives et ceci même pour des cas critiques où la convergence devient extrêmement lente. L'avantage de ces méthodes d'extrapolation est qu'elles utilisent uniquement une suite de vecteurs qui n'est pas forcément convergente, ou qui converge très lentement pour créer une nouvelle suite qui converge plus vite et pouvant admettre une convergence quadratique. De plus, le développement de méthodes redémarrées (ou cycliques) permet de limiter le coût et le stockage. Nous cherchons les méthodes itératives les plus efficaces pour la résolution de ce type d'équation de Riccati. Le problème de convergence de ces méthodes est examiné tout en identifiant les cas critiques correspondant à une convergence très lente. Ensuite, nous appliquons l'extrapolation polynomiale à ces méthodes afin d'améliorer leur convergence. Une tâche importante relative à l'analyse du cas critique et son interprétation a été réalisée. Nous avons utilisé une technique de décalage « shift technique » afin d'éliminer le problème lié à la singularité de la matrice Jacobienne. En résumé, en transformant l'équation de départ avec une technique de « shift » nous évitons le problème de singularité pour la matrice Jacobienne. L'efficacité de l'approche proposée est illustrée à travers plusieurs exemples numériques.
Mots clés : Méthodes d'extrapolation polynomiale, accélération de la convergence, méthodes redémarrées, méthodes itératives, suite de vecteurs, systèmes non linéaires, équation de Riccati algébrique , théorie du transport, une solution minimale positive, une matrice Jacobienne, cas critique.