Scuola Normale Superiore, Pisa — Classe di Scienze

Algorithms for quadratic matrix and vector equations

by
Federico Poloni

xx yyyuary 201x

Thesis submitted for the degree of Doctor of Philosophy

Advisors: prof. Dario A. Bini University of Pisa
prof. Beatrice Meini University of Pisa

Contents

Chapter 1

Introduction

This thesis is devoted to studying algorithms for the solution of a class of quadratic matrix and vector equations appearing, in different forms, in several practical applications.

With the term *quadratic vector (matrix) equation* we denote an equation in which the unknown is a vector (or a matrix), and the equation itself contains only terms of degree at most 2 in the entries of the unknown. Thus, strictly speaking, we are dealing with a system of N equations in N unknowns, each of degree 2. However, in all our algorithms it is useful to consider the unknowns as a whole vector or matrix, and apply the techniques and tools of linear algebra. This allows a richer and more numerics-friendly treatment than abstract commutative algebra tools. Therefore, we favor the approach of considering them a single equation in a vector or matrix unknown.

A common aspect to many of the equations that we consider is the presence of suitable (componentwise) nonnegativity assumptions on the coefficients, which ensure the existence of nonnegative solutions. The solution of interest is the minimal nonnegative one, according again to the componentwise ordering or to a spectral characterization. The properties of the cone

$$\mathbb{R}^n_+ := \{x \in \mathbb{R}^n : x_i \geq 0 \text{ for } i = 1, \ldots, n\},$$

play an important role, together with the Perron–Frobenius theory of positive matrices and M-matrices. Some remarks on this theory and other linear algebra results which are needed along the exposition are recalled in Chapter 2.

A first equation, arising originally from the modeling of Markovian binary trees [BKT08, HLR08], has the form

$$Mx = a + b(x, x), \tag{1.1}$$

where the unknown x and a are in $\mathbb{R}^N_+$, M is an M-matrix, and $b : \mathbb{R}^N_+ \times \mathbb{R}^N_+ \to \mathbb{R}^N_+$ is a bilinear form. This is in fact a very general equation, as many of the quadratic matrix and vector equations appearing in literature can be represented in this form. The analysis of some basic properties and algorithms for this equation, performed in Chapter 3, allows us to introduce, under a unifying point of view, several common aspects of quadratic vector and matrix equations with nonnegativity assumptions. Among them, the minimal solution, the monotonic convergence of basic functional iterations and of Newton's method, and the key concept of criticality.

In Chapter 4, we introduce a new algorithm which applies to the quadratic vector equations (1.1) of the original probability setting [BKT08] and has a completely different behavior from the traditional fixed-point iterations when the problem is close to critical.

In Chapter 5, we treat in more detail the so-called *unilateral quadratic matrix equation* (UQME)

$$AX^2 + BX + C = 0 \tag{1.2}$$

appearing originally in a model in queuing theory [BLM05]. The usual assumptions in this probabilistic setting are that

$$B = -(I - B'), \qquad A, B', C \in \mathbb{R}_+^{n' \times n'}, \qquad (A + B' + C)e = e. \tag{1.3}$$

Equation (1.2) is strictly related to the quadratic eigenvalue problem

$$(\lambda^2 A + \lambda B + C)u = 0. \tag{1.4}$$

The minimal solution and the convergence properties of several algorithms can be characterized in terms of the properties of the roots of (1.4).

An effective algorithm for the solution of (1.2) is the cyclic reduction (CR) [BGN70, BLM05], a quadratically convergent iteration which implements a sort of squaring of the eigenvalues of the original problem. Logarithmic reduction [LR93] (LR) is a closely related iteration arising from a probabilistic idea, which is somewhat simpler to analyze in some respects. In the same Chapter 5, we discuss the relationship between CR/LR and the Newton method for quadratic vector equations.

Another matrix equation which falls in our framework is the nonsymmetric algebraic Riccati equation (NARE)

$$AX + XD = XCX + B, \tag{1.5}$$

with $X, B \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{n \times m}$, $A \in \mathbb{R}^{m \times m}$, $D \in \mathbb{R}^{n \times n}$. In its nonsymmetric version, this equation appears in fluid queues [BOT05], which are a continuous-time version of the models that lead to (1.2). The nonnegativity assumption here is that the matrix

$$\mathcal{M} := \begin{bmatrix} D & -C \\ -B & A \end{bmatrix} \tag{1.6}$$

is a nonsingular or singular irreducible M-matrix.

The NARE (1.5) can be reformulated as an invariant subspace problem

$$\begin{bmatrix} D & -C \\ B & -A \end{bmatrix} \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} (D - CX) \tag{1.7}$$

for the matrix

$$\mathcal{H} := \begin{bmatrix} D & -C \\ B & -A \end{bmatrix} = \mathcal{K}\mathcal{M}, \qquad \text{with } \mathcal{K} := \begin{bmatrix} I_n & 0 \\ 0 & -I_m \end{bmatrix}, \tag{1.8}$$

called *Hamiltonian*. This name is a bit misleading, as the terminology arises from the field of (symmetric) continuous-time algebraic Riccati equations [LR95], which have been widely studied in control theory since 1960. With the notation of (1.5), they represent the case where $m = n$, $D = A^T$, and B and C are symmetric. In this case, the matrix $\mathcal{H} \in \mathbb{R}^{2n \times 2n}$ is a *Hamiltonian matrix*, i.e., it satisfies

$$\mathcal{J}\mathcal{H} = (\mathcal{J}\mathcal{H})^T, \qquad \mathcal{J} := \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}. \tag{1.9}$$

The Structured doubling algorithm (SDA) [And78, GLX06] is an algorithm for the solution of nonsymmetric algebraic Riccati equations that has been recently rediscovered,

understood in terms of eigenvalues of matrix pencils and applied to many different equations. In Chapter 6, we describe this algorithm and study its connection to cyclic reduction, based on the evolution of an approach suggested by Ramaswami [Ram99] on how to transform a NARE into a UQME.

In Part II, we deal with a special case of nonsymmetric algebraic Riccati equation arising from a physical problem in neutron transport theory [JL99]. Given two parameters $0 < c \leq 1$, $0 \leq \alpha < 1$, this equation is (1.5) with parameters $A, B, C, D \in \mathbb{R}^{n\times n}$ given by

$$A = \Delta - eq^T, \qquad B = ee^T, \qquad C = qq^T, \qquad D = \Gamma - qe^T, \tag{1.10}$$

and

$$\begin{aligned} e &= (1, 1, \ldots, 1)^T, \\ q &= (q_1, q_2, \ldots, q_n)^T \qquad \text{with } q_i = \frac{w_i}{2t_i}, \\ \Delta &= \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_n) \quad \text{with } \delta_i = \frac{1}{ct_i(1+\alpha)}, \\ \Gamma &= \operatorname{diag}(\gamma_1, \gamma_2, \ldots, \gamma_n) \quad \text{with } \gamma_i = \frac{1}{ct_i(1-\alpha)}. \end{aligned} \tag{1.11}$$

The quantities $0 < t_n < \ldots < t_2 < t_1 < 1$ and $w_i > 0, i = 1, 2, \ldots, n$, with $\sum_i w_i = 1$, are the nodes and weights of a suitable Gaussian quadrature rule. For more details and for the physical meaning of the two parameters, we refer the reader to [JL99] and to the references therein.

Although the problem can be treated and solved in the framework of (1.5) (in particular, $\mathcal{M}$ is an irreducible M-matrix), there is a richer structure to exploit: $\mathcal{M}$ and $\mathcal{H}$ are diagonal plus rank one matrices, the solution is a Cauchy-like matrix, and most algorithms preserve the Cauchy-like structure along all the iterates. Alternatively, following an idea of L.-Z. Lu [Lu05b], the equation can be rewritten in terms of its Cauchy-like generators [Lu05b] as

$$\begin{cases} u = u \mathbin{.*} (Pv) + e, \\ v = v \mathbin{.*} (\tilde{P}u) + e, \end{cases} \tag{1.12}$$

with $P, \tilde{P} \in \mathbb{R}_+^{n\times n}$ given rank-structured matrices, and unknowns $u, v \in \mathbb{R}_+^n$. Here the operator $.*$ denotes the componentwise (Hadamard) product of two vectors of the same length, with a notation borrowed from Matlab. This equation falls in the same class of nonnegative quadratic vector equations, thus the algorithms introduced in Chapter 3 can be applied productively. In Chapter 8 we introduce Cauchy-like matrices and their properties, and we develop variants of the existing algorithms; in Chapter 9 we discuss the structured form of Newton's method for (1.5) and (1.12) and their mutual relationship. We showed in [BMP09] that SDA and the Ramaswami cyclic reduction algorithm [Ram99] can also be implemented in a structure-preserving way; however, this approach does not seem to give numerically sound results, therefore it is not treated in full detail.

In Part III we turn to the original application of SDA, control theory problems, and describe two new ways to exploit this algorithm in the solution of invariant subspace problems. The structured doubling algorithm is traditionally [CFL05] applied to the symmetric, continuous-time algebraic Riccati equation which arises from a continuous-time control problem after some manipulations. In more detail, optimal control problems appear originally [Son98, Meh91] in the form of an invariant subspace problem for a matrix pencil in the form

$$s\mathcal{E} - \mathcal{A} = \begin{bmatrix} 0 & -sI + A & B \\ sI + A^* & Q & C \\ B^* & C^* & R \end{bmatrix}, \tag{1.13}$$

where the block sizes are such that $A \in \mathbb{C}^{n\times n}$, $R \in \mathbb{C}^{m\times m}$, and $\mathcal{A}$ is Hermitian. When R is nonsingular, the last block row and columns can be eliminated, leading to a reduced invariant subspace problem for a Hamiltonian matrix. Moreover, when the leading square block of the stable invariant subspace is nonsingular, the problem can be further reduced to the form (1.7), which, as we already noted, is equivalent to an algebraic Riccati equation.

This is the approach that is followed in the standard engineering literature on control theory. However, in some cases, the nonsingularity assumptions that we have made do not hold, and thus the algebraic Riccati equation cannot be formed. Different techniques are needed for the solution of this class of control problems. Chapter 10 is devoted to solving a system of equations describing control problems with a singular matrix R, the so-called Lur'e equations. Chapter 11 describes a possible variation of SDA in order to deal more reliably with problems in which a singular leading square block appears in the stable invariant subspace, and thus the Riccati equation cannot be formed.

Part IV deals with the generalization of another quadratic matrix equation. The matrix geometric mean $A \# B$ of two symmetric, positive definite matrices may be defined [Bha07] as the unique symmetric positive definite solution X of the Riccati equation

$$XA^{-1}X = B, \tag{1.14}$$

or directly as

$$A \# B := A(A^{-1}B)^{1/2} = A^{1/2}(A^{-1/2}BA^{-1/2})^{1/2}A^{1/2}. \tag{1.15}$$

There are deep connections between the geometric mean defined here and the Riemannian geometry of positive definite matrices. In several applications, the geometric mean of two matrices arises as a natural candidate to average the results of numerical experiments affected by uncertainty [Moa06]. However, when the matrices to average are more than two, it is not easy to find a sensible generalization maintaining most of the properties that hold for the two-matrix case. A first generalization was proposed in a seminal paper by Ando, Li and Mathias (ALM mean); in Chapter 12 we describe a different one, defined via a natural geometrical approach. Moreover, it turns out that our mean is faster to compute than the original ALM mean, since it is based on an iteration with cubical convergence; on the other hand, the original mean by Ando, Li and Mathias was based on an iteration with linear convergence.

However, the problem of finding a computationally effective matrix geometric mean is far from being solved. Although faster than the original Ando–Li–Mathias iteration, the new iteration has a cost that still grows exponentially with the number of involved matrices. In Chapter 13 we show that, at least under reasonable assumptions, all the means constructed using this approach (composing previously existing matrix means and taking limits) require an amount of computational work that grows exponentially with the number of matrices to average.

At the end of each chapter, we report comments and research lines on the exposed topics. Where appropriate, the results of numerical experiments are presented. In most of them, Matlab is the language of choice; however, in Part II, we switch to Fortran 90 for some tests, since we need tight `for` loops that would perform poorly in the Matlab architecture.

Chapter 2

Linear algebra preliminaries

With the notations R^n and $R^{m\times n}$ we denote respectively the space of n-vectors and $m \times n$ matrices with coefficients in a ring R — or a semi-ring, as in the case of

$$\mathbb{R}_+ := \{x \in \mathbb{R} : x \geq 0\}.$$

The notation I_n, with n often omitted when it is clear from the context, denotes the identity matrix; the zero matrix of any dimension is denoted simply by 0. With e we denote the vector of suitable dimension all of whose entries are 1. The expression $\rho(A)$ stands for the spectral radius of the matrix A.

We make use of some handy matrix notations taken from Fortran 90 and Matlab. When M is a matrix, the symbol $M_{i:j,k:\ell}$ denotes the submatrix formed by rows i to j and columns k to ℓ of M, including extremes. The index i is a shorthand for $i:i$, and : alone is a shorthand for $1:n$, where n is the maximum index allowed for that row/column. A similar notation is used for vectors. When $v \in \mathbb{C}^n$ is a vector, the symbol $\mathrm{diag}(v)$ denotes the diagonal matrix $D \in \mathbb{C}^{n\times n}$ such that $D_{i,i} = v_i$. On the other hand, when $M \in \mathbb{C}^{n\times n}$ is a square matrix, $\mathrm{diag}(M)$ denotes the (column) vector with entries $M_{1,1}, M_{2,2}, \ldots, M_{n,n}$. Finally, we denote with $u .* v$ the componentwise (Hadamard) product of two vectors $u, v \in R^n$, i.e., $w = u .* v \in R^n$ is the vector such that $w_i = u_i v_i$ for each $i = 1, 2, \ldots, n$.

The notation $A \geq 0$ has a double meaning in linear algebra; unfortunately, in this thesis we need both, though in different parts.

- In parts I and II, we write $A \geq B$, where $A, B \in \mathbb{R}^{m\times n}$, to denote $A - B \in \mathbb{R}_+^{m\times n}$, i.e., $A_{i,j} \geq B_{i,j}$ for all valid i, j (componentwise ordering), and similarly with vectors. Writing $A > B$ means that $A_{i,j} > B_{i,j}$ for all i, j.
- In parts III and IV, we write $A \geq B$, only for symmetric matrices in $\mathbb{R}^{n\times n}$, to mean that $A - B$ is positive semidefinite. Similarly, $A > B$ means that $A - B$ is positive definite (positive definite ordering).

2.1 Nonnegative matrices and M-matrices

In this section, the symbols $\geq$ and $>$ denote the componentwise ordering.

A matrix $A \in \mathbb{R}^{n\times n}$ is called *irreducible* when its associated graph is strongly connected, i.e., if the set $\{1, 2, \ldots, n\}$ cannot be partitioned in two disjoint sets S and T such that $A_{s,t} = 0$ whenever $s \in S$, $t \in T$.

The following theorem is at the basis of the theory of (componentwise) nonnegative matrices.

Theorem 2.1 (Perron–Frobenius theorem [BP94a])**.** *Let P be a nonnegative matrix. Then,*

- *$\rho(P)$ is an eigenvalue of P, called its* Perron value
- *P has nonnegative left and right eigenvectors w^T and v corresponding to $\rho(P)$*

If in addition P is irreducible, then

- *$\rho(P)$ is a simple eigenvalue of P*
- *its associated right and eigenvectors w^T and v, called the* left and right Perron vectors, *have strictly positive components. All other left and right eigenvectors have at least one strictly negative entry.*
- *If $Q \geq P$ but $Q \neq P$, then $\rho(Q) > \rho(P)$.*

Corollary 2.2 ([BIMP10])**.** *Let A be an irreducible nonnegative matrix and let $v_1, \ldots, v_n$ be a set of Jordan chains of A. Then there exists only one positive or negative vector among the v_i's and it is a scalar multiple of v.*

A real square matrix Z is said to be a *Z-matrix* if $Z_{ij} \leq 0$ for all $i \neq j$. A Z-matrix is said to be an *M-matrix* if it can be written in the form $sI - P$, where $P \geq 0$ and $s \geq \rho(P)$ and $\rho(\cdot)$ denotes the spectral radius.

The following results are classical, see e.g. [BP94a].

Theorem 2.3. *The following properties hold.*

1. *If Z is a Z-matrix and there exists a vector $v > 0$ such that $Zv \geq 0$, then Z is an M-matrix;*
2. *If Z is a Z-matrix and $Z \geq M$ for an M-matrix $M \neq Z$, then Z is a nonsingular M-matrix.*
3. *A nonsingular Z-matrix Z is an M-matrix if and only if $Z^{-1} \geq 0$.*

Lemma 2.4. *Let M be a nonsingular M-matrix or an irreducible singular M-matrix. Partition M as*

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix},$$

where M_{11} and M_{22} are square matrices. Then M_{11} and M_{22} are nonsingular M-matrices. The Schur complement of M_{11} (or M_{22}) in M is also an M-matrix (singular or nonsingular according to M). Moreover, the Schur complement is irreducible if M is irreducible.

2.2 Sherman–Morrison–Woodbury formula

A matrix identity that is needed in the following is the so-called *Sherman–Morrison–Woodbury (SMW) formula* [GVL96, p. 50].

Lemma 2.5 (Sherman–Morrison–Woodbury formula [GVL96])**.** *Let $D \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{k\times k}$ be nonsingular, and $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{k\times n}$. Then $D - UCV$ is nonsingular if and only if $C^{-1} - VD^{-1}U$ is nonsingular, and it holds that*

$$(D - UCV)^{-1} = D^{-1} + D^{-1}U(C^{-1} - VD^{-1}U)^{-1}VD^{-1}.$$

The following lemma relates the SMW formula and M-matrices.

Lemma 2.6. *Let D, C, U, V be real matrices satisfying the hypotheses of Lemma 2.5, with D and C diagonal and $D, C, U, V \geqslant 0$. Then, $D - UCV$ is a (nonsingular) M-matrix if and only if $C^{-1} - VD^{-1}U$ is a (nonsingular) M-matrix.*

Proof. Let $C^{-1} - VD^{-1}U$ be a nonsingular M-matrix, the SMW formula yields

$$(D - UCV)^{-1} = D^{-1} + D^{-1}U(C^{-1} - VD^{-1}U)^{-1}VD^{-1},$$

and since all terms on the right-hand side are nonnegative, one has $(D - UCV)^{-1} \geq 0$, so $D - UCV$ is a nonsingular M-matrix by Theorem 2.3; the converse is analogous. By a continuity argument, the result can be extended to singular M-matrices. □

2.3 Newton's method

Let $F : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be a continuous map; F is said to be *Fréchet differentiable* if there is a linear operator A such that

$$\lim_{h \to 0} \frac{\|F(x+h) - Fx - Ah\|}{\|h\|} = 0.$$

The map A is denoted by F'_x, and is called the *Fréchet derivative* of F at x.

Let $F : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be a Fréchet differentiable map. The Newton method for the solution of the equation $F(x) = 0$, starting from a point $x_0 \in D$, is defined by

$$x_{k+1} = x_k - (F'_{x_k})^{-1} F(x_k), \tag{2.1}$$

and is well-defined whenever F'_{x_k} is nonsingular for all $k = 0, 1, \ldots$.

The following result is a weaker version of the classical Newton-Kantorovich theorem.

Theorem 2.7 ([OR00])**.** *Let F be Fréchet differentiable on D, and F' be Lipschitz continuous. Let x^* be a zero of F such that $F'(x^*)$ is nonsingular. Then there exists a radius r such that the Newton method* (2.1) *started from any x_0 with $\|x - x_0\| < r$ converges to x^* quadratically.*

Notice that convergence is not guaranteed when $F'(x^*)$ is singular; in fact, pathological cases may arise [GO83].

2.4 Matrix polynomials

We present here a minimal introduction to matrix polynomials and pencils, where we state only the results that are needed in this thesis. A complete theory, including Jordan chains, normal forms and careful definition of eigenvalues at infinity can be found in [GLR82].

A *matrix polynomial* is a polynomial expression in which the coefficients are matrices in $\mathbb{C}^{m \times n}$, i.e.,

$$A(\lambda) = \sum_{i=0}^{d} \lambda^i A_i, \qquad A_i \in \mathbb{C}^{m \times n},$$

where λ is the indeterminate. A value $\widehat{\lambda}$ for which $\det A(\widehat{\lambda}) = 0$ is called an *eigenvalue* of the matrix polynomial. The corresponding vectors u such that $A(\widehat{\lambda})u = 0$ are called *eigenvectors*.

For a polynomial (or, more generally, rational) matrix-valued function $\Phi : D \subseteq \mathbb{C} \to \mathbb{C}^{n\times m}$, we define the normal rank by $\operatorname{normalrank} \Phi = \max_{s\in D} \operatorname{rk} \Phi(s)$.

A matrix polynomial is called *regular* if $m = n$ and $\operatorname{normalrank} A(\lambda) = n$, or, equivalently, if $\det A(\lambda)$ does not vanish for each λ. In this case, $\det A(\lambda)$ is a polynomial of degree at most nd, and thus has at most nd solutions. When the leading coefficient A_d is singular, the degree is strictly less than nd. In this case, we may consider the degree deficiency as *eigenvalues at infinity*. More precisely, if the matrix polynomial

$$\operatorname{rev} A(\lambda) := \sum_{i=0}^{d} \lambda^i A_{d-i}$$

has an eigenpair $(0, u)$, then we call (∞, u) an eigenpair at infinity for the matrix polynomial $A(\lambda)$.

Notice that every similarity transformation of the form

$$A(\lambda) \mapsto RA(\lambda)S, \tag{2.2}$$

with nonsingular $R, S \in \mathbb{C}^{n\times n}$, preserves the eigenvalues of the matrix polynomial. If $S = I$, the (right) eigenvectors are preserved, too.

2.5 Matrix pencils

A degree-1 matrix polynomial is called a *matrix pencil*. We often write matrix pencils in the form $sE - A$. This is to highlight how square matrix pencils are, in many respects, a generalization of the usual matrices: many of the definition related to eigenvalues, Jordan forms and deflating (invariant) subspaces that we give in this section reduce to the usual ones for a matrix A if $E = I$. Thus we can embed square matrices in the space of square matrix pencils by identifying $A \equiv sI - A$.

A canonical form for matrix pencils is given by the following result. For $k \in \mathbb{N}$ we introduce the special matrices $J_k, M_k, N_k \in \mathbb{R}^{k,k}$, $K_k, L_k \in \mathbb{R}^{k-1,k}$ with

$$J_k := \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}, \qquad K_k := \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \end{bmatrix}, \quad L_k := \begin{bmatrix} 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix},$$

$$M_k := \begin{bmatrix} & & 1 & 0 \\ & \iddots & \iddots & \\ 1 & \iddots & & \\ 0 & & & \end{bmatrix}, \quad N_k := \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix}.$$

Theorem 2.8 (Kronecker canonical form (KCF) [Gan98])**.** *For a matrix pencil $sE - A$ with $E, A \in \mathbb{C}^{m\times n}$, there exist nonsingular matrices $U_l \in \mathbb{C}^{m\times m}$, $U_r \in \mathbb{C}^{n\times n}$, such that*

$$U_l(sE - A)U_r = \operatorname{diag}(\mathcal{C}_1(s), \ldots, \mathcal{C}_k(s)), \tag{2.3}$$

where each of the pencils $\mathcal{C}_j(s)$ is of one of the types presented in Table 2.1.

Type	Size	$\mathcal{C}_j(s)$	Parameters
W1	$k_j \times k_j$	$(s-\lambda)I_{k_j} - N_{k_j}$	$k_j \in \mathbb{N}$, $\lambda \in \mathbb{C}$
W2	$k_j \times k_j$	$sN_{k_j} - I_{k_j}$	$k_j \in \mathbb{N}$
W3	$(k_j-1) \times k_j$	$sK_{k_j} - L_{k_j}$	$k_j \in \mathbb{N}$
W4	$k_j \times (k_j-1)$	$sK_{k_j}^T - L_{k_j}^T$	$k_j \in \mathbb{N}$

Table 2.1: Block types in Kronecker canonical form

Type	Size	$\mathcal{D}_j(s)$	Parameters
E1	$2k_j \times 2k_j$	$\begin{bmatrix} 0_{k_j,k_j} & (\lambda-s)I_{k_j} - N_{k_j} \\ (\overline{\lambda}+s)I_{k_j} - N_{k_j}^T & 0_{k_j,k_j} \end{bmatrix}$	$k_j \in \mathbb{N}$, $\lambda \in \mathbb{C}^+$
E2	$k_j \times k_j$	$\epsilon_j((-is-\mu)J_{k_j} + M_{k_j})$	$k_j \in \mathbb{N}, \mu \in \mathbb{R}, \epsilon_j = \pm 1$
E3	$k_j \times k_j$	$\epsilon_j(isM_{k_j} + J_{k_j})$	$k_j \in \mathbb{N}, \epsilon_j = \pm 1$
E4	$(2k_j-1)\times$ $(2k_j-1)$	$\begin{bmatrix} 0_{k_j-1,k_j-1} & -sK_{k_j} + L_{k_j} \\ sK_{k_j}^T + L_{k_j}^T & 0_{k_j,k_j} \end{bmatrix}$	$k_j \in \mathbb{N}$

Table 2.2: Block types in even Kronecker canonical form

The blocks of type W1 are Jordan blocks associated with an eigenvalue λ of $sE - A$. Blocks of type W2 correspond to infinite eigenvalues. Blocks of type W3 and W4 appear only for non-regular pencils.

A pencil $sE - A$ is called *even* if $E = -E^*$ and $A = A^*$. A special modification of the KCF for even matrix pencils is presented in[Tho76].

Theorem 2.9 (even Kronecker canonical form (EKCF)[Tho76])**.** *For an even matrix pencil $sE - A$ with $E, A \in \mathbb{C}^{n\times n}$, there exists a nonsingular $U \in \mathbb{C}^{n\times n}$ such that*

$$U^*(sE - A)U = \operatorname{diag}(\mathcal{D}_1(s), \ldots, \mathcal{D}_k(s)), \tag{2.4}$$

where each of the pencils $\mathcal{D}_j(s)$ is of one of the types presented in Table 2.2.

The numbers ϵ_j in the blocks of type E2 and E3 are called the *block signatures*. The blocks of type E1 contain complex conjugate pairs $(\lambda, -\overline{\lambda})$ of eigenvalues. The blocks of type E2 and E3 correspond respectively to the purely imaginary and infinite eigenvalues. Blocks of type E4 appear only in non-regular pencils and are a combination of two blocks of type W3 and W4.

A generalization of the concept of invariant subspace applies naturally to matrix pencils. A subspace $\mathcal{U} = \operatorname{Span} U$, with $U \in R^{n\times r}$ having full column rank, is called an r-dimensional *(right) deflating subspace* for the matrix pencil $sE - A$, $A, E \in R^{m\times n}$, if there are $V \in R^{n\times r}$, and $\widetilde{A}, \widetilde{E} \in R^{r\times r}$ such that

$$(A - sE)U = V(\widetilde{A} - s\widetilde{E}).$$

Two pencils $sE_1 - A_1$ and $sE_2 - A_2$ are called *right-similar* if there is a nonsingular $L \in \mathbb{C}^{m\times m}$ such that

$$L(sE_1 - A_1) = sE_2 - A_2.$$

In this case, they have the same eigenvalues, right deflating subspaces and Jordan chains. Similarly, a square matrix pencil $sE - A$ is said to be *right-similar* to a matrix M of the same size if $L(sE - A) = sI - M$ for some nonsingular L, i.e., $M = E^{-1}A$.

2.6 Indefinite product spaces

Especially in Part III, we deal with the space $\mathbb{R}^{2n}$ equipped with the indefinite form $\omega(x, y) = x^T \mathcal{J} y$ defined by the matrix $\mathcal{J}$ introduced in (1.9). Notice that $\mathcal{J}^{-1} = \mathcal{J}^T = -\mathcal{J}$.

A real *Hamiltonian matrix* is one that is skew-symmetric with respect to this form, i.e., $\mathcal{H}^T\mathcal{J} = -\mathcal{J}\mathcal{H}$. Notice that this definition is essentially the same as the one given in (1.9). This reduces to imposing that the matrix $\mathcal{H}$ has the block form (1.8) with $A = D^T$, $B = B^T$, $C = C^T$. A real *symplectic matrix* is one that is orthogonal with respect to $\mathcal{J}$, i.e., $\mathcal{S}^T J\mathcal{S} = \mathcal{J}$.

A matrix pencil $\mathcal{A} - \lambda\mathcal{E}$ is called *symplectic* if $\mathcal{A}\mathcal{J}\mathcal{A}^T = \mathcal{E}\mathcal{J}\mathcal{E}^T$.

A subspace $\mathcal{U} = \operatorname{Span} U$ is called *Lagrangian* if $U^T\mathcal{J}U = 0$, or equivalently, $\omega(u, v) = 0$ for all $u, v \in \mathcal{U}$.

Real Hamiltonian matrices and real even pencils have the so-called *Hamiltonian eigensymmetry*, i.e., if $\lambda \in \mathbb{C}$ is an eigenvalue, then so are $\overline{\lambda}$, $-\lambda$, $-\overline{\lambda}$. That is, the spectrum is symmetric with respect to the real and imaginary axes. Eigenvalues with nonzero real and imaginary part come in quadruples; purely real or imaginary eigenvalues come in pairs, except for the eigenvalues 0 and ∞ (in the case of pencils), which need not be paired.

Real symplectic matrices and pencils have the so-called *symplectic eigensymmetry*, i.e., if $\lambda \in \mathbb{C}$ is an eigenvalue, then so are $\overline{\lambda}, 1/\lambda, 1/\overline{\lambda}$. That is, the spectrum is symmetric with respect to the real axis and to the unit circle. Non-unimodular, non-real eigenvalues come in quadruples; real and unimodular eigenvalues come in pair, except for 1 and -1. In the case of pencils, this pairing is respected provided that we stick to the usual conventions of arithmetic involving (complex) infinity.

2.7 Möbius transformations and Cayley transforms

Let $a, b, c, d \in \mathbb{C}$ be given such that $ad - bc \neq 0$. The map from $\mathbb{P}(\mathbb{C}) = \mathbb{C} \cup \{\infty\}$ to itself defined by $f(z) := \frac{az+b}{cz+d}$ is called a *Möbius transformation*. Möbius transformations are projective automorphisms of the Riemann sphere.

Möbius transformations can be extended to matrices: if $cM + dI$ is nonsingular, then $f(M) := (cM + d)^{-1}(aM + b) = (cM + dI)^{-1}(aM + bI)$. As can be seen by reducing M to Jordan or Schur form, λ is an eigenvalue of M if and only if $f(\lambda)$ is an eigenvalue of $f(M)$. Left and right eigenvectors and Jordan chains are preserved, as follows from the results in [Hig08].

A more natural extension, which removes the nonsingularity assumption, involves matrix pencils:

$$f(\mathcal{A} - \lambda\mathcal{E}) = (a\mathcal{A} + b\mathcal{E}) - \lambda(c\mathcal{A} + d\mathcal{E}).$$

This map transforms eigenvalues (including those at infinity) according to $\lambda \mapsto f(\lambda)$, and preserves left and right Jordan chains and deflating subspaces.

A special class of Möbius transformation is given by *Cayley transforms*. The Cayley transform with parameter $\gamma \in \mathbb{R} \setminus \{0\}$ is the map

$$\mathcal{C}_\gamma(z) = \frac{z-\gamma}{z+\gamma}. \tag{2.5}$$

The following properties follow easily from the definition.

Lemma 2.10. *The following properties hold for any Cayley transform $\mathcal{C} = \mathcal{C}_\gamma$.*

1. $\mathcal{C}(0) = 1$
2. $\mathcal{C}(\infty) = -1$
3. *$\mathcal{C}$ maps the real axis (including ∞) onto itself*
4. *$\mathcal{C}$ maps the imaginary axis onto the unit circle. If $\gamma > 0$, then the right half-plane is mapped onto the unit disk, and the left half-plane to its outside; if $\gamma < 0$, the left half-plane is mapped onto the unit disk, and the right half-plane to its outside.*
5. *$\mathcal{C}$ sends a matrix/pencil with Hamiltonian eigensymmetry to a matrix/pencil with symplectic eigensymmetry.*

The Cayley transform preserves left and right eigenvectors, while transforms the associated eigenvalues according to the map $\mathcal{C} : \lambda \mapsto \frac{\lambda-\gamma}{\lambda+\gamma}$, $\lambda \in \mathbb{C} \cup \infty$. In particular, Kronecker blocks of size k for λ are mapped to Kronecker blocks of size k for $\mathcal{C}(\lambda)$. This follows for example from [Hig08, Theorem 1.36].

2.8 Control theory terminology

A complete introduction to control theory is outside of the scope of this thesis; we refer the reader to [Son98] for more detail of this topic. However, in Part III we deal with invariant subspace problems arising from control theory applications. While our exposition is self-contained and need not deal with the engineering background of the problem, it is useful to introduce some of the notation used in the field.

Most control theory definitions exist in two variants, one relative to linear recurrence systems (*discrete-time* problems) and one to linear differential equations (*continuous-time* problems). The difference is that in the former case the stability region of interest is the open unit disk $\{z \in \mathbb{C} : |z| < 1\}$, while in the latter it is the open left half-plane $\{z \in \mathbb{C} : \Re(z) < 0\}$. Thus most definitions appear in two variants. For example, a matrix is called *stable* if all its eigenvalues lie inside the stability region; this means that a continuous-time stable (*c-stable*) matrix has all its eigenvalues inside the left half plane, while a discrete-time stable (*d-stable*) matrix has all its eigenvalues inside the unit disk. The prefixes d- and c- are used when it is necessary to distinguish between the two cases.

An eigenvalue is called *stable* if it lies inside the stability region, *unstable* if it lies outside its closure, and *critical* if it lies on its border. A matrix (or pencil) is called *semi-stable* if all its eigenvalues are stable or critical. Conversely, it is *unstable* (*semi-unstable*) if all its eigenvalues are unstable (unstable or critical).

The *stable space* of a matrix (pencil) is the invariant subspace (deflating subspace) spanned by the Jordan chains relative to stable eigenvalues; similarly, the *unstable space* is the invariant (deflating) subspace relative to unstable eigenvalues.

Due to the properties in Lemma 2.10, a Cayley transform with parameter $\gamma < 0$ maps c-stable eigenvalues to d-stable eigenvalues, and thus, roughly speaking, transforms all the

stated continuous-time conditions into the corresponding discrete-time conditions. If $\gamma > 0$, then it also switches the stable and unstable regions.

A matrix pair (A, B), with $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times r}$, is called *controllable* if $\operatorname{rk}\begin{bmatrix}\lambda I - A & B\end{bmatrix} = n$ for all $\lambda \in \mathbb{C}$, and *stabilizable* if this holds only for λ with $\Re\lambda \geq 0$. A matrix pair (C, A) is called *observable* if (A^T, C^T) is controllable, and *detectable* if (A^T, C^T) is stabilizable. Notice that these definitions do not depend on the choice of continuous- or discrete-time setting.

2.9 Eigenvalue splittings

We speak of a *splitting* of the eigenvalues of a matrix when referring on how they divide between the stable, critical and unstable region of the complex plane.

A matrix or pencil of size $m+n$ has an (m, n) *splitting* if there exist $r_1, r_2 \in \mathbb{N}$ such that:

- its stable space has dimension $m - r_1$;
- its unstable space has dimension $n - r_2$;
- its critical space has dimension $r_1 + r_2$.

The splitting is called *proper* if at least one among r_1 and r_2 is zero. When this happens, we can identify uniquely two disjoint semi-stable and semi-unstable invariant (or deflating) subspaces of dimension respectively m and n: if $r_1 = 0$, then the critical space is considered to be part of the semi-unstable subspace, while if r_2 is zero, then it is considered to be part of the semi-unstable subspace. We call these two subspaces the *canonical* semi-stable and semi-unstable subspaces.

The splitting is called *partial splitting* if $r_1 = r_2$, and the lengths k_j of its Jordan chains relative to critical eigenvalues (which must sum up to $2r$ if the two previous properties hold) are all even. In this case, we define the *canonical* semi-stable (resp. semi-unstable) subspace as the invariant subspace spanned by all the Jordan chains relative to stable (resp. unstable) eigenvalues, plus the first $k_j/2$ vectors from each critical chain.

Symplectic and Hamiltonian matrices and pencils have respectively a d- and c-partial splitting. Under suitable assumptions [LR95] which are satisfied in most control theory applications, the canonical semi-stable and semi-unstable subspaces of a symplectic or Hamiltonian matrix or pencil are Lagrangian.

In the case of a d-splitting, we call the *dominance factor* of a splitting the ratio

$$\nu := \max \frac{|\lambda_s|}{|\lambda_u|} \leq 1, \tag{2.6}$$

where the maximum is taken over all the eigenvalues λ_s belonging to the canonical d-semi-stable subspace, and all the eigenvalues λ_u belonging to the canonical d-semi-unstable subspace. The dominance factor ν is small when there is a large spectral gap between the semi-stable and semi-unstable subspace; it is strictly less than 1 in the case of a proper splitting, and equal to 1 in case of a partial splitting. The convergence properties of many of the algorithms for matrix equations that we expose are related to the dominance factor.

Part I

Quadratic vector and matrix equations

Chapter 3

Quadratic vector equations

3.1 Introduction

In this chapter, we aim to study in an unified fashion several quadratic vector and matrix equations with nonnegativity hypotheses. Specific cases of these problems have been studied extensively in the past by several authors. For references to the single equations and results, we refer the reader to the following sections, in particular Section 3.3. Many of the results appearing here have already been proved for one or more of the single instances of the problems, resorting to specific characteristics of the problem. In some cases the proofs we present here are mere rewritings of the original proofs with a little change of notation to adapt them to our framework, but in some cases we are effectively able to remove some hypotheses and generalize the results by abstracting the specific aspects of each problem.

It is worth noting that Ortega and Rheinboldt [OR00, Chapter 13], in a 1970 book, treat a similar problem in a far more general setting, assuming only the monotonicity and operator convexity of the involved operator. Since their hypotheses are far more general than the ones of our problem, the obtained results are less precise than the one we are reporting here. Moreover, all of their proofs have to be adapted to our case, since our map $F(x)$ is operator concave instead of convex.

3.2 General problem

We are interested in solving the equation (1.1) (quadratic vector equation, QVE), where $M \in \mathbb{R}^{N\times N}$ is a nonsingular M-matrix, $a, x \in \mathbb{R}^N_+$, and b is a nonnegative vector bilinear form, i.e., a map $b : \mathbb{R}^N_+ \times \mathbb{R}^N_+ \to \mathbb{R}^N_+$ such that $b(v,\cdot)$ and $b(\cdot,v)$ are linear maps for each $v \in \mathbb{R}^n_+$. The map b can be represented by a tensor B_{ijk}, in the sense that $b(x,y)_k = \sum_{i,j=1}^n B_{ijk}x_iy_j$. It is easy to prove that $x \le y, z \le w$ implies $b(x,z) \le b(y,w)$. If A is a nonsingular M-matrix, $A^{-1}B$ denotes the tensor representing the map $(x,y) \mapsto A^{-1}b(x,y)$. Note that, here and in the following, we do not require that b be symmetric (that is, $b(x,y) = b(y,x)$ for all x, y): while in the equation only the quadratic form associated with b is used, in the solution algorithms there are often terms of the form $b(x,y)$ with $x \neq y$. Since there are multiple ways to extend the quadratic form $b(x,x)$ to a bilinear map $b(x,y)$, this leaves more freedom in defining the actual solution algorithms.

We are only interested in nonnegative solutions $x^* \in \mathbb{R}^N_+$; in the following, when referring to solutions of (1.1) we always mean *nonnegative* solutions only. A solution x^* of (1.1) is called *minimal* if $x^* \le y^*$ for any other solution y^*.

Later on, we give a necessary and sufficient condition for (1.1) to have a minimal solution.

3.3 Concrete cases

E1: Markovian binary trees in [BKT08, HLR08], the equation (1.1), with the assumption that e is a solution, arises from the study of Markovian binary trees.

E2: Lu's simple equation in [Lu05b, Lu05a], the equation (1.12) arises from a special Riccati equation appearing in a neutron transport problem. By setting $w := [u^T\, v^T]^T$ as the new unknown, the equation takes the form (1.1).

E3: Nonsymmetric algebraic Riccati equation in [GL00], the equation (1.5), where the matrix $\mathcal{M}$ defined in (1.6) is a nonsingular or singular irreducible M-matrix, is studied. Vectorizing everything, we get

$$(I \otimes A + D^T \otimes I)\,\mathrm{vec}(X) = \mathrm{vec}(B) + \mathrm{vec}(XCX),$$

which is in the form (1.1) with $N = mn$.

E4: Unilateral quadratic matrix equation in several queuing problems [BLM05], the equation (1.2) with assumptions (1.3) is considered. Vectorizing everything, we fall again in the same class of equations, with $N = n'^2$: in fact, since $B'e \le e$, $Be \ge 0$ and thus B is an M-matrix.

To ease the notation in the cases E3 and E4, in the following we set $x_k = \mathrm{vec}(X_k)$, $d = \max(m, n)$ for E3 and $d = n'$ for E4.

3.4 Minimal solution

Existence of the minimal solution

It is clear by considering the scalar case ($N = 1$) that (1.1) may have no real solutions. The following additional condition allows us to prove their existence.

Assumption 3.1. There are a positive linear functional $l : \mathbb{R}^N \to \mathbb{R}^t$ and a vector $z \in \mathbb{R}^t_+$ such that for any $x \in \mathbb{R}^N_+$, the property $l(x) \le z$ implies $l(M^{-1}(a + b(x, x))) \le z$.

Theorem 3.1 ([Pol10b])**.** *Equation* (1.1) *has at least one solution if and only if Assumption 3.1 holds. Among its solutions, there is a minimal one.*

Proof. Let us consider the iteration

$$x_{k+1} = M^{-1}\left(a + b(x_k, x_k)\right), \tag{3.1}$$

starting from $x_0 = 0$. Since M is an M-matrix, we have $x_1 = M^{-1}a \ge 0$. It is easy to see by induction that $x_k \le x_{k+1}$:

$$x_{k+1} - x_k = M^{-1}(b(x_k, x_k) - b(x_{k-1}, x_{k-1})) \ge 0$$

since b is nonnegative. We prove by induction that $l(x_k) \le z$. The base step is clear: $l(0) = 0 \le z$; the inductive step is simply Assumption 3.1. Thus the sequence x_k is nondecreasing and bounded from above by $l(Mx_k) \le z$, and therefore it converges. Its limit x^* is a solution to (1.1).

On the other hand, if (1.1) has a solution s, then we may choose $l = I$ and $z = s$; now, $x \le s$ implies $M^{-1}(a + b(x, x)) \le M^{-1}(a + b(s, s)) = s$, thus Assumption 3.1 is satisfied with this choices.

For any solution s, we may prove by induction that $x_k \le s$:

$$s - x_{k+1} = a + b(s,s) - a - b(x_k, x_k) \ge 0.$$

Therefore, passing to the limit, we get $x^* \le s$. □

Taylor expansion

Let $F(x) := Mx - a - b(x,x)$. Since the equation is quadratic, the following expansion holds.

$$F(y) = F(x) + F'_x(y-x) + \frac{1}{2}F''_x(y-x, y-x), \tag{3.2}$$

where $F'_x(w) = Mw - b(x,w) - b(w,x)$ is the (Fréchet) derivative of F and $F''_x(w,w) = -2b(w,w) \le 0$ is its second (Fréchet) derivative. Notice that F''_x is nonpositive and does not depend on x.

The following theorem is a straightforward extension to our setting of the argument in [GL00, Theorem 3.2].

Theorem 3.2 ([Pol10b])**.** *If $x^* > 0$, then F'_{x^*} is an M-matrix.*

Proof. Let us consider the fixed point iteration (3.1). By a theorem on fixed-point iterations [KVZ+72], one has

$$\limsup \sqrt[k]{\|x^* - x_k\|} \le \rho(\mathcal{G}'_{x^*}), \tag{3.3}$$

where $\mathcal{G}'_{x^*}$ is the Fréchet derivative of the iteration map

$$\mathcal{G}(x) := M^{-1}(a + b(x,x)),$$

that is,

$$\mathcal{G}'_{x^*}(y) = M^{-1}(b(x^*, y) + b(y, x^*)).$$

In fact, if $x^* > 0$, equality holds in (3.3). Let $e_k := x^* - x_k$. We have $e_{k+1} = P_k e_k$, where

$$P_k := M^{-1}(b(x^*, \cdot) + b(\cdot, x_k))$$

are nonnegative matrices. The matrix sequence P_k is nondecreasing and $\lim_{k\to\infty} P_k = \mathcal{G}'_{x^*}$. Thus for any $\varepsilon > 0$ we may find an integer l such that

$$\rho(P_m) \ge \rho(\mathcal{G}_{x^*}) - \varepsilon, \quad \forall m \ge l.$$

We have

$$\begin{aligned}\limsup \sqrt[k]{\|x^* - x_k\|} &= \limsup \sqrt[k]{\|P_{k-1} \dots P_l \dots P_0 x^*\|} \\ &\ge \limsup \sqrt[k]{\left\|P_l^{k-l} P_0^l x^*\right\|}.\end{aligned}$$

Since $x^* > 0$, we have $P_0^l x^* > 0$ and thus $P_0^l x^* > c_l e$ for a suitable constant c_l. Also, it holds that $\left\|P_l^{k-l}\right\| = \left\|P_l^{k-l} v_{k,l}\right\|$ for a suitable $v_{k,l} \ge 0$ with $\|v_{k,l}\| = 1$, and this implies

$$\begin{aligned}\limsup \sqrt[k]{\|x^* - x_k\|} &= \limsup \sqrt[k]{c_l \left\|P_l^{k-l} e\right\|} \\ &\ge \limsup \sqrt[k]{c_l \left\|P_l^{k-l} v_{k,l}\right\|} \\ &= \limsup \sqrt[k]{c_l \left\|P_l^{k-l}\right\|} \\ &= \rho(P_l) \ge \rho(\mathcal{G}_{x^*}) - \varepsilon.\end{aligned}$$

Since ε is arbitrary, this shows that equality holds in (3.3).

From the convergence of the sequence x_k, we get thus

$$\rho(M^{-1}(b(x^*,\cdot)+b(\cdot,x^*)))\leq 1,$$

which implies that $M-b(x^*,\cdot)-b(\cdot,x^*)$ is an M-matrix. □

Corollary 3.3. *From part 2 of Theorem 2.3, we promptly obtain that $F'(x)$ is an M-matrix for all $x\leq x^*$.*

Concrete cases

We may prove Assumption 3.1 for all the examples E1–E4. E1 is covered by the following observation.

Lemma 3.4. *If there is a vector $y\geq 0$ such that $F(y)\geq 0$, then Assumption 3.1 holds.*

Proof. In fact, we may take the identity map as l and y as z. Clearly $x\leq y$ implies $M^{-1}(a+b(x,x))\leq M^{-1}(a+b(y,y))\leq y$. □

As for E2, it follows from the reasoning in [Lu05b] that a solution to the specific problem is $u=Xq+e$, $v=X^Tq+e$, where X is the solution of an equation of the form E3; therefore, E2 follows from E3 and Lemma 3.4. An explicit but rather complicate bound to the solution is given in [Jua01].

The case E3 is treated in [Guo01, Theorem 3.1]. Since $\mathcal{M}$ in (1.6) is a nonsingular or singular M-matrix, there are vectors $v_1,v_2>0$ and $u_1,u_2\geq 0$ such that $Dv_1-Cv_2=u_1$ and $Av_2-Bv_1=u_2$. Let us set $l(x)=Xv_1$ and $z=v_2-A^{-1}u_2$. We have

$$\begin{aligned}(AX_{k+1}+X_{k+1}D)v_1 =&(X_kCX_k+B)v_1\\ \leq& X_kCv_2+Av_2-u_2\leq XDv_1+Av_2-u_2.\end{aligned}$$

Since $X_{k+1}Dv_1\geq X_kDv_1$ (monotonicity of the iteration), we get $X_{k+1}v_1\leq v_2-A^{-1}u_2$, which is the desired result.

The case E4 is similar. It suffices to set $l(x)=\mathrm{vec}^{-1}(x)e$ and $z=e$:

$$X_{k+1}e=(I-B)^{-1}(A+CX_k^2)e\leq(I-B)^{-1}(Ae+Ce)\leq e,$$

since $(A+C)e=(I-B)e$

3.5 Functional iterations

Definition and convergence

We may define a functional iteration for (1.1) by choosing a splitting $b=b_1+b_2$ such that $b_i\geq 0$ and a splitting $M=Q-P$ such that Q is an M-matrix and $P\geq 0$. We then have the iteration

$$(Q-b_1(\cdot,x_k))x_{k+1}=a+Px_k+b_2(x_k,x_k). \tag{3.4}$$

Theorem 3.5 ([Pol10b])**.** *Suppose that the equation* (1.1) *has a minimal solution $x^*>0$. Let x_0 be such that $0\leq x_0\leq x^*$ and $F(x_0)\leq 0$ (e.g., $x_0=0$). Then:*

1. *$Q-b_1(\cdot,x_k)$ is nonsingular for all k, i.e., the iteration* (3.4) *is well-defined.*

2. $x_k \le x_{k+1} \le x^*$, *and* $x_k \to x^*$ *as* $k \to \infty$.

3. $F(x_k) \le 0$ *for all* k.

Proof. Let $J(x) := Q - b_1(\cdot, x)$ and $g(x) := a + Px + b_2(x, x)$. It is clear from the nonnegativity constraints that J is nonincreasing (i.e., $x \le y \Rightarrow J(x) \ge J(y)$) and g is nondecreasing (i.e., $x \le y \Rightarrow g(x) \le g(y)$).

The matrix $J(x)$ is a Z-matrix for all $x \ge 0$. Moreover, since $J(x^*)x^* = g(x^*) \ge 0$, $J(x^*)$ is an M-matrix by Theorem 2.3 and thus, by the same theorem, $J(x)$ is a nonsingular M-matrix for all $x \le x^*$.

We first prove by induction that $x_k \le x^*$. This shows that the iteration is well-posed, since it implies that $J(x_k)$ is an M-matrix for all k. Since $g(x^*) = J(x^*)x^* \le J(x_k)x^*$ by inductive hypothesis, (3.4) implies

$$J(x_k)(x^* - x_{k+1}) \ge g(x^*) - g(x_k) \ge 0;$$

thus, since $J(x_k)$ is an M-matrix by inductive hypothesis, we deduce that $x^* - x_{k+1} \ge 0$.

We prove by induction that $x_k \le x_{k+1}$. For the base step, since we have $F(x_0) \le 0$, and $J(x_0)x_0 - g(x_0) \le 0$, thus $x_1 = J(x_0)^{-1}g(x_0) \ge x_0$. For $k \ge 1$, it holds that

$$J(x_{k-1})(x^{k+1} - x_k) \ge J(x_k)x_{k+1} - J(x_{k-1})x_k = g(x_k) - g(x_{k-1}) \ge 0,$$

thus $x_k \le x_{k+1}$. The sequence x_k is monotonic and bounded above by x^*, thus it converges. Let x be its limit; by passing (3.4) to the limit, we see that x is a solution. But since $x \le x^*$ and x^* is minimal, it must be the case that $x = x^*$.

Finally, for each k we have

$$F(x_k) = J(x_k)x_k - g(x_k) \le J(x_k)x_{k+1} - g(x_k) = 0. \qquad \square$$

Theorem 3.6 ([Pol10b])**.** *Let* f *be the map defining the functional iteration* (3.4)*, i.e.,* $f(x_k) = J(x_k)^{-1}g(x_k) = x_{k+1}$*. Let* $\hat{J}$*,* $\hat{g}$*,* $\hat{f}$ *be the same maps as* J*,* g*,* f *but for the special choice* $b_2 = 0$*,* $P = 0$*. Then* $\hat{f}^k(x) \ge f^k(x)$*, i.e., the functional iteration with* $b_2 = 0$*,* $P = 0$ *has the fastest convergence among all those defined by* (3.4)*.*

Proof. It suffices to prove that $\hat{f}(y) \ge \hat{f}(x)$ for all $y \ge x$, which is obvious from the fact that $\hat{J}$ is nonincreasing and $\hat{g}$ is nondecreasing, and that $\hat{f}(x) \ge f(x)$, which follows from the fact that $\hat{J}(x) \le J(x)$ and $\hat{g}(x) \ge g(x)$. $\square$

Corollary 3.7. *Let*

$$x_{k+1}^{GS} = J(y_k)^{-1}g_k, \tag{3.5}$$

where y_k *is a vector such that* $x_k \le y_k \le x_{k+1}$*, and* g_k *a vector such that* $g(x_k) \le g_k \le g(x_{k+1})$*. It can be proved with the same arguments that* $x_{k+1} \le x_{k+1}^{GS} \le x^*$*. This implies that we can perform the iteration in a "Gauss–Seidel" fashion: if in some place along the computation an entry of* x_k *is needed, and we have already computed the same entry of* x_{k+1}*, we can use that entry instead. It can be easily shown that* $J(x_k)^{-1}g(x_k) \le J(y_k)^{-1}g_k$*, therefore the Gauss–Seidel version of the iteration converges faster than the original one.*

Remark 3.8. The iteration (3.4) depends on b as a bilinear form, while (1.1) and its solution depend only on b as a quadratic form. Therefore, different choices of the bilinear form b lead to different functional iterations for the same equation. Since for each iterate of each functional iteration both $x_k \le x^*$ and $F(x_k) \le 0$ hold (thus x_k is a valid starting point for a new functional iteration), we may safely switch between different functional iterations at every step.

Concrete cases

For E1, the algorithm called *depth* in [BKT08] is given by choosing $P = 0$, $b_2 = 0$. The algorithm called *order* in the same paper is obtained with the same choices, but starting by the bilinear form $\tilde{b}(x, y) := b(y, x)$ obtained by switching the arguments of b. The algorithm called *thicknesses* in [HLR08] is given by performing alternately one iteration of each of the two above methods.

For E2, Lu's simple iteration [Lu05b] and the algorithm NBJ in [BGL08] can be seen as the basic iteration (3.1) and $P = 0$, $b_2 = 0$. The algorithm NBGS in the same paper is a Gauss–Seidel-like variant.

For E3, the fixed point iterations in [GL00] are given by $b_2 = b$ and different choices of P. The iterations in [JC93] are the one given by $b_2 = 0, P = 0$ and a Gauss–Seidel-like variant.

For E4, the iterations in [BLM05, Chapter 6] can also be reinterpreted in our framework.

3.6 Newton's method

Definition and convergence

We may define the Newton method for the equation (1.1) as

$$F'_{x_k}(x_{k+1} - x_k) = -F(x_k). \tag{3.6}$$

Alternatively, we may write

$$F'_{x_k} x_{k+1} = a - b(x_k, x_k).$$

Also notice that

$$-F(x_k) = b(x_{k-1} - x_k, x_{k-1} - x_k). \tag{3.7}$$

Theorem 3.9 ([Pol10b])**.** *If $x^* > 0$, the Newton method* (3.6) *starting from $x_0 = 0$ is well-defined, and the generated sequence x_k converges monotonically to x^*.*

Proof. First notice that, since F'_{x^*} is an M-matrix, by Theorem 3.2, F'_x is a nonsingular M-matrix for all $x \leq x^*$, $x \neq x^*$.

We prove by induction that $x_k \leq x_{k+1}$. We have $x_1 = M^{-1}a \geq 0$, so the base step holds. From (3.7), we get

$$F'_{x_{k+1}}(x_{k+2} - x_{k+1}) = b(x_{k+1} - x_k, x_{k+1} - x_k) \geq 0,$$

thus, since $F'_{x_{k+1}}$ is a nonsingular M-matrix, $x_{k+2} \geq x_{k+1}$, which completes the induction proof.

Moreover, we may prove by induction that $x_k < x^*$. The base step is obvious, the induction step is

$$\begin{aligned} F'_{x_k}(x^* - x_{k+1}) =& Mx^* - b(x_k, x^*) - b(x^*, x_k) - a + b(x_k, x_k) \\ =& b(x^* - x_k, x^* - x_k) > 0. \end{aligned}$$

The sequence x_k is monotonic and bounded from above by x^*, thus it converges; by passing (3.6) to the limit we see that its limit must be a solution of (1.1), hence x^*. □

Concrete cases

Newton methods for E1, E2, and E3 appear respectively in [HLR08], [Lu05a] and [GL00], with more restrictive hypotheses which hold true in the special applicative cases. As far as we know, the more general hypothesis $x^* > 0$ first appeared here. In particular, in [GL00] (and later [Guo01]) the authors impose that $x_1 > 0$; [HLR08] impose that F'_{x^*} is an M-matrix, which is true in their setting because of probabilistic assumptions; and in the setting of [Lu05a], $x_1 > 0$ is obvious. The Newton method is usually not considered for E4 due to its high computational cost.

3.7 Modified Newton method

Recently Hautphenne and Van Houdt [HVH10] proposed a different version of Newton's method for E1 that has a better convergence rate than the traditional one. Their idea is to apply the Newton method to the equation

$$G(x) = x - (M - b(\cdot, x))^{-1} a, \tag{3.8}$$

which is equivalent to (1.1).

Theoretical properties

Let us set for the sake of brevity $R_x := M - b(\cdot, x)$. The Jacobian of G is

$$G'_x = I - R_x^{-1} b(R_x^{-1} a, \cdot).$$

As for the original Newton method, it is a Z-matrix, and a nonincreasing function of x. It is easily seen that G'_{x^*} is an M-matrix. The proof in Hautphenne and Van Houdt [HVH10] is of probabilistic nature and cannot be extended to our setting; we provide here a different one. We have

$$G'(x^*) = R_{x^*}^{-1} \left(M - b(\cdot, x^*) - b(R_{x^*}^{-1} a, \cdot)\right) = R_{x^*}^{-1} \left(M - b(\cdot, x^*) - b(x^*, \cdot)\right);$$

the quantity in parentheses is F'_{x^*}, an M-matrix, thus there is a vector $v > 0$ such that $F'_{x^*} v \geq 0$, and therefore $G'_{x^*} v = R_{x^*}^{-1} F'_{x^*} v \geq 0$. This shows that G'_x is an M-matrix for all $x \leq x^*$ and thus the modified Newton method is well-defined. The monotonic convergence is easily proved in the same fashion as for the traditional method.

The following result holds.

Theorem 3.10 ([HVH10])**.** *Let $\tilde{x}_k$ be the iterates of the modified Newton method and x_k those of the traditional Newton method, starting from $\tilde{x}_k = x_k = 0$. Then $\tilde{x}_k - x_k \geq 0$.*

The proof in Hautphenne and Van Houdt [HVH10] can be adapted to our setting with minor modifications.

Concrete cases

Other than for E1, its original setting, the modified Newton method can be applied successfully for the other concrete cases of quadratic vector equations. In order to outline here the possible benefits of this strategy, we ask for the reader's indulgence and refer forward to the results on the structure of Newton's method which are presented in Chapter 9.

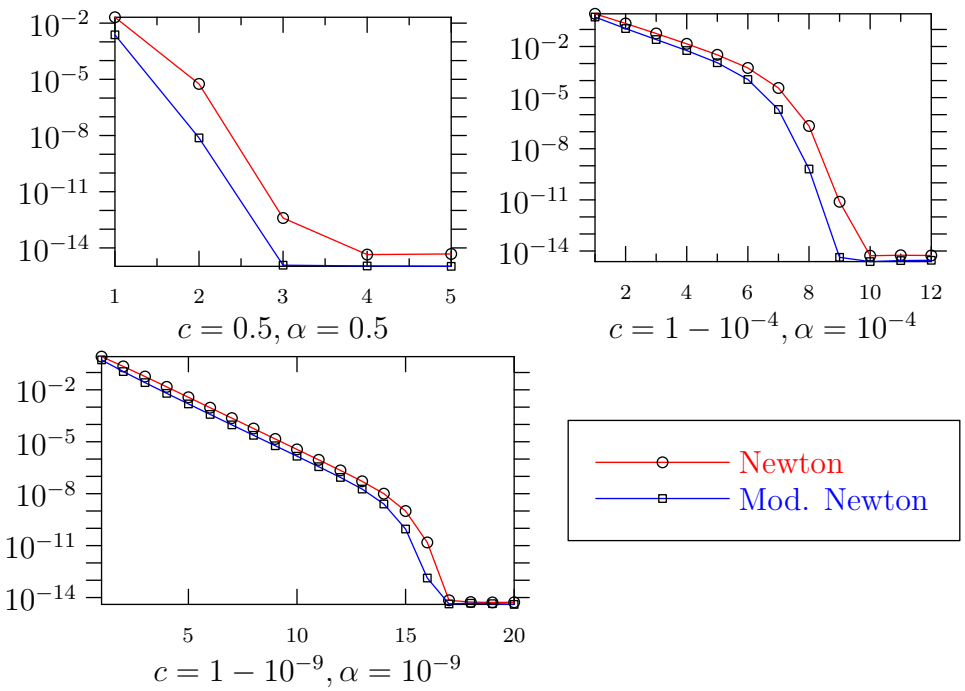


Figure 3.1: Convergence history of the two Newton methods for E2 for several values of the parameters α and c. The plots show the residual Frobenius norm of Equation (1.1) vs. the number of iterations

For E2, let us choose the bilinear map b as

$$b([u_1 v_1], [u_2 v_2]) := [u_1 \circ (P v_2), v_1 \circ (\tilde{P} u_2)].$$

It is easily seen that $b(\cdot, x)$ is a diagonal matrix. Moreover, as in the traditional Newton method, the matrix $b(x, \cdot)$ has a special structure (Trummer-like structure [BIP08], also in Chapter 9) which allows a fast inversion with $O(n^2)$ operations per step. Therefore the modified Newton method can be implemented with a negligible overhead ($O(n)$ ops per step on an algorithm that takes $O(n^2)$ ops per step) with respect to the traditional one, and increased convergence rate.

We have performed some numerical experiments on the modified Newton method for E2; as can be seen in Figure 3.1, the modified Newton method does indeed converge faster to the minimal solution, and this allows one to get better approximations to the solution with the same number of steps. The benefits of this approach are limited, however; in the numerical experiments performed, this modified Newton method never saved more than one iteration with respect to the customary one.

For E3 and E4, the modified Newton method leads to similar equations to the traditional one (continuous- and discrete-time Sylvester equations), but requires additional matrix inversions and products; that is, the overhead is of the same order $O(d^3)$ of the cost of the Newton step. Therefore it is not clear whether the improved convergence rate makes up for the increase in the computational cost.

3.8 Positivity of the minimal solution

Role of the positivity

In many of the above theorems, the hypothesis $x^* > 0$ is required. Is it really necessary? What happens if it is not satisfied?

In all the algorithms we have exposed, we worked with only vectors x such that $0 \leq x \leq x^*$. Thus, if x^* has some zero entry, we may safely replace the problem with a smaller one by projecting the problem on the subspace of all vectors that have the same zero pattern as x^*: i.e., we may replace the problem with the one defined by

$$\hat{a} = \Pi a,\ \hat{M} = \Pi M \Pi^T,\ \hat{b}(x, y) = \Pi b(\Pi^T x, \Pi^T y),$$

where Π is the orthogonal projector on the subspace

$$W = \{x \in \mathbb{R}^N : x_i = 0 \text{ for all } i \text{ such that } x_i^* = 0\}, \tag{3.9}$$

i.e., the linear operator that removes the entries known to be zero from the vectors. Performing the above algorithms on the reduced vectors and matrices is equivalent to performing them on the original versions, provided the matrices to invert are nonsingular. Notice, though, that both functional iterations and Newton-type algorithms may break down when the minimal solution is not strictly positive. For instance, consider the problem

$$a = \begin{bmatrix} \frac{1}{2} \\ 0 \end{bmatrix},\ M = I_2,\ b\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = \begin{bmatrix} \frac{1}{2} x_1 y_1 \\ K x_1 y_2 \end{bmatrix},\ x^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

For suitable choices of the parameter K, the matrices to be inverted in the functional iterations (excluding obviously (3.1)) and Newton's methods are singular; for large values of K, none of them are M-matrices. However, the nonsingularity and M-matrix properties still hold for their restrictions to the subspace W defined in (3.9). It is therefore important to consider the positivity pattern of the minimal solution in order to get efficient algorithms.

Computing the positivity pattern

By considering the functional iteration (3.1), we may derive a method to infer the positivity pattern of the minimal solution in time $O(N^3)$. Let us denote by e_t the t-th vector of the canonical basis, and $e_S = \sum_{s \in S} e_s$ for any set $S \in \{1, \ldots, N\}$. We report the algorithm as Algorithm 1.

Theorem 3.11 ([Pol10b])**.** *The above algorithm runs in at most $O(N^3)$ operations and computes the set $\{h \in \{1, \ldots, N\} : x_h^* > 0\}$.*

Proof. For the implementation of the working sets, we use the simple approach to keep in memory two vectors $S, T \in \{0, 1\}^N$ and set to 1 the components corresponding to the indices in the sets. With this choice, insertions and membership tests are $O(1)$, loops are easy to implement, and retrieving an element of the set costs at most $O(N)$.

Let us first prove that the running time of the algorithm is at most $O(N^3)$. If we precompute a PLU factorization of M, each subsequent operation $M^{-1}v$, for $v \in \mathbb{R}^N$, costs $O(n^2)$. The first **for** loop runs in at most $O(N)$ operations. The body of the **while** loop runs at most N times, since an element can be inserted into S and T no more than once (S never decreases). Each of its iterations costs $O(N^2)$, since evaluating $b(e_t, e_S)$ is equivalent

Algorithm 1: Compute the positivity pattern of the solution x^*

input: a, M, b
output: $S = \{i : x^*_i > 0\}$
  $S \leftarrow \emptyset$ {entries known to be positive}
  $T \leftarrow \emptyset$ {entries to check}
  $a' \leftarrow M^{-1}a$
  for $i = 1$ to N **do**
    if $a'_i > 0$ **then**
      $T \leftarrow T \cup \{i\}$; $S \leftarrow S \cup \{i\}$
    end if
  end for
  while $T \neq \emptyset$ **do**
    $t \leftarrow$ some element of T
    $T \leftarrow T \setminus \{t\}$
    $u \leftarrow M^{-1}\left(b(e_S, e_t) + b(e_t, e_S)\right)$ {or only its positivity pattern}
    for $i \in \{1, \dots, N\} \setminus S$ **do**
      if $u_i > 0$ **then**
        $T \leftarrow T \cup \{i\}$; $S \leftarrow S \cup \{i\}$
      end if
    end for
  end while
  return S

to computing the matrix-vector product between the matrix $(B_{tij})_{i,j=1,\dots,N}$ and e_S, and similarly for $b(e_S, e_t)$.

The fact that the algorithm computes the right set may not seem obvious at first sight. Since the sequence x_k is increasing, if one entry in x_k is positive, then it is positive for all iterates x_h, $h > k$. When does an entry of x_k become positive? The positive entries of x_1 are those of $M^{-1}a$; then, an entry t of x_{k+1} is positive if either the corresponding entry of x_k was positive or two entries r, s of x_k were positive and $B_{rst} > 0$. Thus, an entry in x^* is positive if and only if we can find a sequence S_i of subsets of $\{1, \dots, N\}$ such that:

- $S_0 = \{h \in \{1, \dots, N\} : (M^{-1}a)_h > 0\}$;
- $S_{i+1} = S_i \cup \{t_i\}$, and there are two elements $r, s \in S_i$ such that $B_{rst_i} > 0$.

For each element u of $\{h \in \{1, \dots, N\} : x^*_h > 0\}$, we may prove by induction on the length of its minimal sequence S_i that it eventually gets into S (and T). In fact, suppose the last element of the sequence is S_l. Then, by inductive hypothesis, all the elements of S_{l-1} eventually get into S and T. All of them are removed from T at some step of the algorithm. When the last one is removed, the **if** condition triggers and the element u is inserted into T. Conversely, if u is an element that gets inserted into S, then by considering the values of S at the successive steps of the algorithm we get a valid sequence $\{S_i\}$. □

It is a natural question to ask whether for the cases E3 and E4 it is possible to use the special structure of M and b in order to develop a similar algorithm with running time $O(d^3)$, that is, the same as the cost per step of the basic iterations. Unfortunately, we were unable to go below $O(d^4)$. It is therefore much less appealing to run this algorithm as a preliminary step before, since its cost is likely to outweigh the cost of the actual solution. However, we remark that the strict positiveness of the coefficients is usually a property of

the problem rather than of the specific matrices involved, and can often be solved in the model phase before turning to the actual computations. An algorithm such as the above one would only be needed in an "automatic" subroutine to solve general instances of the problems E3 and E4.

3.9 Other concrete cases

In Bini *et al.* [BLM03], the matrix equation

$$X + \sum_{i=1}^{d} A_i X^{-1} D_i = B - I$$

appears, where $B, A_i, D_i \geq 0$ and the matrices $B + D_j + \sum_{i=1}^{d} A_i$ are stochastic. The solution $X = T - I$, with $T \geq 0$ minimal and sub-stochastic, is sought. Their paper proposes a functional iteration and Newton's method. By setting $Y = -X^{-1}$ and multiplying both sides by Y, we get

$$(I - B)Y = I + \sum A_i Y D_i Y,$$

which is again in the form (1.1). It is easy to see that Y is nonnegative whenever T is substochastic, and Y is minimal whenever T is.

The paper considers two functional iterations and the Newton method; all these algorithm are expressed in terms of X instead of Y, but they essentially coincide with those exposed in this chapter.

3.10 Conclusions and research lines

We presented in this chapter a novel unified approach to the analysis of a family of quadratic vector and matrix equations [Pol10b]. This helps pinpointing which are the minimal hypotheses needed to derive the presented results, and in particular the role of the strict positivity of x^*. In some cases, such as in Theorem 3.9, we can weaken the hypotheses of existing theorems; in other cases, this unified approach allows adapting the algorithms designed for one of such equations to the others, with good numerical results.

There are many open questions that could lead to better theoretical understanding of this class of equations and better solution algorithms.

A first question is whether $x \leq x^*$ implies $F(x) \leq 0$, and under which conditions the converse holds. If an easy criterion is found, then one could safely implement overrelaxation in many of the above algorithms, which would lead to faster convergence.

There are a wide theory and several numerical methods for E3 and E4 that rely on the specific matrix structure of the two equations; convergence properties and characteristics of the solutions are understood in terms of the spectral structure of the involved matrices. Some of these results are exposed in the following chapters. Is there a way to extend this theory to include all quadratic vector equations? What is the right way to generalize spectral properties?

In particular, it would be extremely interesting to find a useful expression of (1.1) in terms of an invariant subspace, as in (1.4) for E4 and (1.7) for E3. Applying recursively the techniques used in [MP09] could yield an expression of the type we are looking for, but with dimensions growing exponentially with N.

On the other hand, finding a more feasible invariant subspace formulation for (1.1) seems to be a challenging or even impossible problem. In fact, if we drop the nonnegativity

assumptions, (1.1) can represent basically any system of N simultaneous quadratic equations. In finite fields, similar problems are known to be NP-hard [CKPS00]. An invariant subspace formulation of polynomial size would probably lead to a polynomial-time algorithm for the finite-field version of this problem, which seems too much to ask for.

Another interesting question is whether we can generalize this approach to the positive-definite ordering on symmetric matrices. This would lead to the further unification of the theory of a large class of equations, including the algebraic Riccati equations appearing in control theory [LR95]. A lemma proved by Ran and Reurings [RR02, Theorem 2.2] could replace the first part of Theorem 2.3 in an extension of the results of this chapter to the positive definite ordering.

Chapter 4

A Perron vector iteration for QVEs

4.1 Applications

Equation (1.1), with the additional conditions

$$M = I, \qquad a + b(e, e) = e \tag{4.1}$$

(i.e, the vector e is a solution, though not necessarily the minimal one), arises from the study of a class of branching processes known as Markovian binary trees. These processes model a population composed of a number of individuals, each of which may be in a state $\varphi \in \{1, 2, \ldots, n\}$. The individuals evolve independently and have state-dependent probabilities of reproducing or dying. Here reproducing means that an individual in state i splits into two individuals in state j and k respectively; the probability of this event is represented by the term b_{ijk} in the equation, while the probability of an individual in state i dying is a_i.

It can be proved with probabilistic means [BOT05] that the minimal solution x^* represents the extinction probability of the colony; i.e., x_i is the probability that the colony generated by a single individual in state i eventually gets extinct.

The binary tree is classified according to the spectral radius of the nonnegative matrix

$$R := b(e, \cdot) + b(\cdot, e).$$

Theorem 4.1 ([AN04])**.** *If assumption* (4.1) *hold for* (1.1)*, then we are in one of the following cases.*

subcritical case $\rho(R) < 1$*, and* $x^* = e$*.*

critical case $\rho(R) = 1$*, and* $x^* = e$*.*

supercritical case $\rho(R) > 1$*, and* $x^* \leq e$*,* $x^* \neq e$*.*

4.2 Assumptions on the problem

Besides (4.1), we assume in this chapter that $x^* > 0$, motivated by the discussion in Section 3.8, and that we are in the supercritical case (otherwise, $x^* = e$ and there is nothing to compute).

A further assumption is that R is irreducible. This is equivalent to imposing that $F'(x^*) = I - b(x^*, \cdot) - b(\cdot, x^*)$, which is an M-matrix according to Theorem 3.2, is irreducible, since irreducibility is only determined by the nonnegativity pattern of $b(\cdot, \cdot)$.

Moreover, we may assume that $e^T b(e-x^*, e-x^*) > 0$; otherwise $b(e-x^*, e-x^*) = 0$ and the problem is trivial since it becomes a linear problem.

4.3 The optimistic equation

If we set $x = e - y$, by using (4.1) and the bilinearity of the operator $b(\cdot,\cdot)$, then (1.1) can be rewritten as

$$y = b(y, e) + b(e, y) - b(y, y). \tag{4.2}$$

The trivial solution is $y = 0$, which corresponds to $x = e$. We are interested in the nontrivial solution $0 < y^* < e$, which gives the sought solution $x^* = e - y^*$.

In the probabilistic interpretation of Markovian binary trees, x^* is the extinction probability, thus $y^* = e - x^*$ is the survival probability, i.e., y_i^* is the probability that a colony starting from a single individual in state i does not become extinct in a finite time. For this reason, we refer to (4.2) as to the *optimistic equation*.

Notice that (4.2) admits the following probabilistic interpretation. The term $b(y, e)$ represents the probability that the original individual $\mathcal{M}$ (for "mother") spawns an offspring $\mathcal{F}$ (for "first-born"), and after that the colony generated by the further offsprings of $\mathcal{M}$, excluding $\mathcal{F}$, survives. The term $b(e, y)$ represents the probability that $\mathcal{M}$ spawns $\mathcal{F}$, and the colony generated by $\mathcal{F}$ survives. The term $b(y, y)$ represents the probability that $\mathcal{M}$ spawns $\mathcal{F}$, and after that both their colonies survive. Thus (4.2) follows by the well-known *inclusion-exclusion* principle

$$\mathbb{P}[\mathcal{M} \text{ or } \mathcal{F} \text{ survives}] = \mathbb{P}[\mathcal{M} \text{ survives}] + \mathbb{P}[\mathcal{F} \text{ survives}] - \mathbb{P}[\text{both } \mathcal{M} \text{ and } \mathcal{F} \text{ survive}],$$

where $\mathbb{P}[X]$ denotes the probability of the event X.

Equation (4.2) can be rewritten as

$$y = H_y y \tag{4.3}$$

where

$$H_y = b(\cdot, e) + b(e, \cdot) - b(y, \cdot). \tag{4.4}$$

Notice that H_y is the sum of a fixed matrix and a matrix that depends linearly on y. Therefore the quadratic operator on the right-hand side of (4.2) is "factored" as the product of a matrix which depends on y, and y.

An important property is that H_y is a nonnegative irreducible matrix, whenever $y < e$. Therefore, by the Perron–Frobenius theorem, if $y < e$, H_y has a positive eigenvalue $\lambda_y = \rho(H_y)$, the so-called *Perron value*, and to λ_y corresponds a positive eigenvector w_y, unique up to a multiplicative constant, the so-called *Perron vector*, so that $H_y w_y = \lambda_y w_y$. Therefore the sought solution y^* can be interpreted as the vector $0 < y^* < e$ such that $\rho(H_{y^*}) = 1$ and y^* is a Perron vector of H_{y^*}.

It is worth pointing out that this interpretation of y^* in terms of the Perron vector allows to keep away from the trivial solution $y = 0$ of (4.3), since the Perron vector has strictly positive elements.

The formulation of the quadratic vector equation in terms of the Perron vector allows to design a new algorithm for its solution.

4.4 The Perron iteration

If we set up a fixed-point iteration or a Newton method for y based on (4.2), we get the traditional fixed-point iterations and Newton methods for MBTs [HLR08], since what we

have done is simply a linear change of variables. Instead, we exploit the fact that y^* is a Perron vector of the nonnegative irreducible matrix H_{y^*} (compare (4.3)).

To this purpose we introduce the operator

$$u = \mathrm{PV}(X)$$

which returns the Perron vector u of the irreducible nonnegative matrix X.

Thus, we can devise a fixed-point iteration to compute the solution y^* by defining the sequence of vectors

$$y_{k+1} = \mathrm{PV}(H_{y_k}), \qquad k = 0, 1, 2, \ldots, \tag{4.5}$$

starting from an initial approximation y_0. In order to define uniquely the sequence $\{y_k\}$, we need to impose a normalization for the Perron vector, which is uniquely defined up to a multiplicative constant. A possible choice for the normalization is imposing that the residual of (4.2) is orthogonal to a suitable vector $w \in \mathbb{R}^n$, i.e.,

$$w^T(y_{k+1} - b(y_{k+1}, e) - b(e, y_{k+1}) + b(y_{k+1}, y_{k+1})) = 0. \tag{4.6}$$

Clearly, this normalization is consistent with the solution of (4.2). We choose w as the left Perron vector of the matrix $b(\cdot, e) + b(e, \cdot)$; the rationale for this choice is discussed in Section 4.5.

Given a Perron vector u of H_{y_k}, the equation to compute the normalization factor α such that $y_{k+1} = \alpha u$ satisfies (4.6) reduces to

$$\alpha w^T u = \alpha w^T b(u, e) + \alpha w^T b(e, u) - \alpha^2 w^T b(u, u),$$

whose only non-zero solution is

$$\alpha = -\frac{w^T(u - b(u, e) - b(e, u))}{w^T b(u, u)}.$$

Notice that the solution $\alpha = 0$ corresponds to the trivial solution $y = 0$ ($x = e$), which we want to avoid.

The $\mathrm{PV}(\cdot)$ operator is defined on the set of irreducible nonnegative matrices. If $y < e$, then the matrix H_y is nonnegative irreducible, therefore the sequence y_k generated by (4.5) is well defined if $y_k < e$ for any k.

In Section 4.5 we show that the iteration (4.5) is locally convergent. Therefore, if y_0 is quite close to y^*, one can expect that $y_k < e$ for any k. In the case where H_{y_k} is not a nonnegative irreducible matrix, we can define y_{k+1} as an eigenvector corresponding to the eigenvalue of H_{y_k} having maximal real part. We call *maximal eigenvector* this eigenvector. Clearly if H_{y_k} is a nonnegative irreducible matrix, the maximal eigenvector is the Perron vector. We see in Section 4.6 that this concern is not necessary in practice.

As a starting approximation y_0 we may choose the null vector. For close to critical problems, where y^* is close to zero, this choice should guarantee the convergence, according to the results of Section 4.5.

The resulting iterative process is summarized in Algorithm 2.

4.5 Convergence analysis of the Perron iteration

In this section, we show that the Perron iteration (4.5) is locally convergent, and its convergence is linear. Moreover, the convergence speed gets faster as the problem gets closer to critical.

Algorithm 2: The Perron iteration

input: b of the QVE (1.1) associated to a supercritical Markovian binary tree
output: the minimal solution x^*

```
k ← 0
y_0 ← 0
w ← the Perron vector of b(e,·) + b(·,e)
while ‖H_{y_k} y_k − y_k‖_1 ≥ ε do
    u ← the maximal eigenvector of H_{y_k}
    Compute the normalization factor α = −w^T(u−b(u,e)−b(e,u)) / w^T b(u,u)
    y_{k+1} ← αu
    k ← k + 1
end while
return x = e − y_k
```

Derivatives of eigenvectors

It is well known [Wil88] that the eigenvalues and eigenvectors of a matrix are analytical functions of the matrix entries in a neighborhood of a simple eigenpair. The following formula gives an analytical expression of their first derivatives.

Theorem 4.2 ([MS88, Theorem 1])**.** *Let $A = A(z)$, $\lambda = \lambda(z)$, $u = u(z)$ be a matrix, eigenvalue and associated eigenvector depending on a parameter $z \in \mathbb{C}$. Let us suppose that $\lambda(z_0)$ is simple and $A'(z_0)$, $\lambda'(z_0)$, $u'(z_0)$ each exist. Let $w = w(z)$ be another vector such that $w'(z_0)$ exists and let $\sigma(u, w)$ be a function whose value is a real scalar constant for all z. Let σ_1^H and σ_2^H be the partial gradients of $\sigma(\cdot, \cdot)$ seen as a function respectively of its first and second vector argument only.*

If $\sigma_1^H u \neq 0$ for $z = z_0$, then the derivative u' of u at $z = z_0$ is given by

$$u' = \frac{\sigma_1^H (A - \lambda I)^{\#} A' u - \sigma_2^H w'}{\sigma_1^H u} u - (A - \lambda I)^{\#} A' u.$$

Here $X^{\#}$ denotes the so-called *group inverse* of a singular matrix X, i.e., the inverse of X in the maximal multiplicative subgroup containing X. We refer the reader to [MS88] for more details on group inverses.

In fact, very little is needed on group inverses, and the formula can be modified slightly in order to replace it with the Moore–Penrose pseudoinverse $X^{\dagger}$, which is a more canonical tool in matrix computations.

Theorem 4.3 ([MP10])**.** *With the same hypotheses as Theorem 4.2, let $v(z)$ be the left eigenvector of $A(z)$ corresponding to the eigenvalue $\lambda(z)$. If $\sigma_1^H u \neq 0$ for $z = z_0$, then the derivative u' of u at $z = z_0$ is given by*

$$u' = \frac{\sigma_1^H (A - \lambda I)^{\dagger} (A' - \lambda' I) u - \sigma_2^H w'}{\sigma_1^H u} u - (A - \lambda I)^{\dagger} (A' - \lambda' I) u, \tag{4.7}$$

with $\lambda' = \frac{v^H A' u}{v^H u}$.

Proof. The proof is a minor modification of the original proof [MS88] of Theorem 4.2. By differentiating the identity $Au = \lambda u$ we get $A'u + Au' = \lambda' u + \lambda u'$, i.e.,

$$(A - \lambda I) u' = -(A' - \lambda' I) u. \tag{4.8}$$

By left-multiplying everything by v^H, and noting that $v^H A = \lambda v^H$, we get the required expression for the eigenvalue derivative λ'. Moreover, since u is a simple eigenvector at $z = z_0$, the kernel of $(A - \lambda I)$ is span(u). Thus from (4.8) we can determine u' up to a scalar multiple of u:

$$u' = -(A - \lambda I)^\dagger (A' - \lambda' I)u + \delta u. \tag{4.9}$$

We now use the normalization condition $\sigma(u, w) = k$ to determine the value of δ. By differentiating it, we get

$$\sigma_1^H(u, w)u' + \sigma_2^H(u, w)w' = 0. \tag{4.10}$$

Plugging (4.9) into (4.10) yields a linear equation for δ. □

Jacobian of the Perron iteration

The Perron iteration is a fixed-point iteration for the function $F(y) := PV(H_y)$, where the function $u = PV(X)$ returns the Perron vector u of the nonnegative irreducible matrix X, normalized such that $w^T(u - H_u u) = 0$, where w is a fixed positive vector. We can use Theorem 4.3 to compute the Jacobian of this map F.

Theorem 4.4 ([MP10])**.** *Let y be such that H_y is nonnegative and irreducible. Let $u = F(y)$, and let v be such that $v^T H_y = \lambda v^T$, where $\lambda = \rho(H_y)$. Then the Jacobian of the map F at y is*

$$JF_y = \left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(H_y - \lambda I)^\dagger \left(I - \frac{uv^T}{v^T u}\right) b(\cdot, u) \tag{4.11}$$

where

$$\sigma_1^T = w^T(I - b(e - u, \cdot) - b(\cdot, e - u)).$$

Proof. We compute first the directional derivative of F at y along the direction a. To this purpose, let us set $y(z) := y + az$, for any $z \in \mathbb{C}$, and $A(z) = H_y$. We have

$$A'(z) = \frac{d}{dz} H_y = -b(a, \cdot).$$

Moreover, set

$$\sigma(u, w) = w^T(u - b(e, u) - b(u, e) + b(u, u)),$$

where $w(z) = w$ for each z (so that $w' = 0$). The partial gradient of $\sigma(\cdot, \cdot)$ with respect to the first argument is $\sigma_1^T = w^T(I - b(e - u, \cdot) - b(\cdot, e - u))$. Plugging everything into (4.7), we get

$$\begin{aligned}
u' =& \frac{\sigma_1^T(A - \lambda I)^\dagger(A' - \lambda' I)u}{\sigma_1^T u} u - (A - \lambda I)^\dagger (A' - \lambda' I)u \\
=& -\left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(A - \lambda I)^\dagger \left(A' - \frac{v^T A' u}{v^T u} I\right) u \\
=& -\left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(A - \lambda I)^\dagger \left(I - \frac{uv^T}{v^T u}\right) A' u \\
=& -\left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(A - \lambda I)^\dagger \left(I - \frac{uv^T}{v^T u}\right) (-b(a, u)) \\
=& \left(I - \frac{u\sigma_1^T}{\sigma_1^T u}\right)(A - \lambda I)^\dagger \left(I - \frac{uv^T}{v^T u}\right) b(\cdot, u) a.
\end{aligned}$$

From this expression for the directional derivative, it is immediate to recognize that the Jacobian is (4.11). □

Local convergence of the iteration

The fixed-point iteration $y_{k+1} = F(y_k)$ is locally convergent in a neighborhood of y^* if and only if the spectral radius of JF_{y^*} is strictly smaller than 1. First notice that it makes sense to compute the Jacobian using (4.11) in a neighborhood of the solution y^*. In fact, $\sigma_1^T y^* = w^T(y^* - b(e-y^*, y^*) - b(y^*, e-y^*)) = w^T b(y^*, y^*)$ and the latter quantity is positive as $w > 0$ and $b(y^*, y^*) \gneq 0$, as stated in Section 4.2. Moreover, since $\lambda = 1$ is a simple eigenvalue, the left and right eigenvectors $v = v^*$ and $u = y^*$ cannot be orthogonal.

By evaluating (4.11) at $y = y^*$, we get

$$JF_{y^*} = \left(I - \frac{y^* \sigma_1^{*T}}{w^T b(y^*, y^*)}\right) A^\dagger \left(I - \frac{y^* v^{*T}}{v^{*T} y^*}\right) b(\cdot, y^*), \tag{4.12}$$

where we have set $\sigma_1^{*T} = w^T(I - b(e-y^*, \cdot) - b(\cdot, e-y^*))$ and $A = (b(e-y^*, \cdot) + b(\cdot, e) - I)$.

Let us try to understand what happens to the spectral radius $\rho(JF_{y^*})$ when the problem is close to critical.

Theorem 4.5 ([MP10])**.** *Let $b_t(\cdot,\cdot)$, $t \in [0,1]$ be an analytical one-parameter family of Markovian binary trees, which is supercritical for $t \in [0,1)$ and critical for $t = 1$, and let us denote with an additional subscript t the quantities defined above for this family of problems. Let us suppose that $R_t := b_t(e,\cdot) + b_t(\cdot, e)$ is irreducible for every $t \in [0,1]$, and let $\rho(JF_{y_t^*,t})$ be the spectral radius of the Jacobian of the Perron iteration as defined in* (4.12)*. Then*

$$\lim_{t\to 1} \rho(JF_{y_t^*,t}) = \left|1 - \frac{\widehat{v}_1^T b_1(\widehat{y}_1, \widehat{y}_1)}{w^T b_1(\widehat{y}_1, \widehat{y}_1)} \frac{w^T \widehat{y}_1}{\widehat{v}_1^T \widehat{y}_1}\right|$$

where $\widehat{y}_1$ and $\widehat{v}_1^T$ are left and right Perron vectors of R_1. As a special case, if the vector w is a scalar multiple of $\widehat{v}_1$, the limit is 0.

Proof. Let us define $\widehat{y}_t$ as the Perron vector of $H_{y_t^*,t}$ normalized so that $\|\widehat{y}_t\|_1 = 1$, and similarly $\widehat{v}_t^T$ as the left Perron vector of the same matrix, normalized so that $\|\widehat{y}_t\|_1 = 1$. Since $H_{y_t^*,t}$ is irreducible, its left and right Perron vectors are analytical functions of t, and thus $\widehat{y}_t$ and $\widehat{v}_t$ converge to $\widehat{y}_1$ and $\widehat{v}_1$ respectively. We have $H_{y_1^*,1} = H_{0,1} = R_1$. Notice that $\sigma_{1,t}^{*,T} \to w^T(I - R_1)$, and that

$$A_t^\dagger = (b(e-y^*,\cdot) + b(\cdot,e) - I)^\dagger \to (R_1 - I)^\dagger.$$

Moreover, since $\widehat{y}_1$ and $\widehat{v}_1$ span the right and left kernel of $I - R_1$, we have

$$(I - R_1)(I - R_1)^\dagger = I - \frac{\widehat{y}_1 \widehat{v}_1^T}{\widehat{v}_1^T \widehat{y}_1}.$$

Additionally, we make use of the relation $\rho(AB) = \rho(BA)$, valid for any A and B such that AB and BA are square matrices, in the first and second-to-last step of the following computation.

Putting all together, we get

$$\begin{aligned}
\rho(JF_{y_t^*,t}) =& \rho\left(\left(I - \frac{y_t^* \sigma_{1,t}^{*T}}{w^T b_t(y_t^*, y_t^*)}\right) A_t^\dagger \left(I - \frac{y_t^* v_t^{*T}}{v_t^{*T} y_t^*}\right) b_t(\cdot, y_t^*)\right) \\
=& \rho\left(\left(b_t(\cdot, y_t^*) - \frac{b_t(y_t^*, y_t^*)\sigma_{1,t}^{*T}}{w^T b_t(y_t^*, y_t^*)}\right) A_t^\dagger \left(I - \frac{y_t^* v_t^{*T}}{v_t^{*T} y_t^*}\right)\right) \\
=& \rho\left(\left(b_t(\cdot, y_t^*) - \frac{b_t(\widehat{y}_t, \widehat{y}_t)\sigma_{1,t}^{*T}}{w^T b_t(\widehat{y}_t, \widehat{y}_t)}\right) A_t^\dagger \left(I - \frac{\widehat{y}_t \widehat{v}_t^T}{\widehat{v}_t^T \widehat{y}_t}\right)\right).
\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}\lim_{t\to 1}\rho(JF_{y_t^*,t}) =&\rho\left(-\frac{b_1(\widehat{y}_1,\widehat{y}_1)w^T(I-R_1)}{w^Tb_1(\widehat{y}_1,\widehat{y}_1)}(R_1-I)^\dagger\left(I-\frac{\widehat{y}_1\widehat{v}_1^T}{\widehat{v}_1^T\widehat{y}_1}\right)\right)\\ =&\rho\left(\frac{b_1(\widehat{y}_1,\widehat{y}_1)w^T}{w^Tb_1(\widehat{y}_1,\widehat{y}_1)}\left(I-\frac{\widehat{y}_1\widehat{v}_1^T}{\widehat{v}_1^T\widehat{y}_1}\right)^2\right)\\ =&\rho\left(\frac{b_1(\widehat{y}_1,\widehat{y}_1)w^T}{w^Tb_1(\widehat{y}_1,\widehat{y}_1)}\left(I-\frac{\widehat{y}_1\widehat{v}_1^T}{\widehat{v}_1^T\widehat{y}_1}\right)\right)\\ =&\rho\left(\frac{1}{w^Tb_1(\widehat{y}_1,\widehat{y}_1)}w^T\left(I-\frac{\widehat{y}_1\widehat{v}_1^T}{\widehat{v}_1^T\widehat{y}_1}\right)b_1(\widehat{y}_1,\widehat{y}_1)\right)\\ =&\left|1-\frac{\widehat{v}_1^Tb_1(\widehat{y}_1,\widehat{y}_1)}{w^Tb_1(\widehat{y}_1,\widehat{y}_1)}\frac{w^T\widehat{y}_1}{\widehat{v}_1^T\widehat{y}_1}\right|.\end{aligned}$$

□

For the normalization condition, the above result suggests taking w as the left Perron vector of $b(e,\cdot)+b(\cdot,e)$. Indeed, this choice guarantees the local convergence of the Perron iteration for close-to-critical problems. Moreover we point out that, even though the convergence is linear, the speed of convergence increases as the MBT gets closer to critical; in particular, the convergence is superlinear in the critical case.

4.6 Numerical experiments

We compared the Perron iteration (PI) with the Newton method (NM) [HLR08] and with the *thicknesses* algorithm (TH) [HLR09]. As stated before, TH and PI are linearly convergent algorithms, while NM is a quadratically convergent one. All the experiments were performed using Matlab 7 (R14) on an Intel Xeon 2.80Ghz bi-processor.

We applied the algorithms to the two test cases reported in [HLR08]. The first one (E1) is an MBT of size $n = 9$ depending on a parameter λ, which is critical for $\lambda \approx 0.85$ and supercritical for larger values of λ. The second one (E2) is a MBT of size $n = 3$ depending on a parameter λ, which is critical for $\lambda \approx 0.34$ and $\lambda \approx 0.84$, and supercritical for the values in this interval.

The only noteworthy issue in the implementation of PI is the method used for the computation of the maximal eigenvector. The classical methods are usually optimized for matrices of much larger size; however, here we deal with matrices of size $n = 3$ and $n = 9$, for which the complexity constants matter. We compared several candidates (`eigs`, `eig`, the power method, a power method accelerated by repeated squaring of the matrix), and found that in our examples the fastest method to find the maximal eigenvector is computing the full eigenvector basis with `[V,Lambda]=eig(P)` and then selecting the maximal eigenvector. The picture should change for problems of larger size: `eig` takes $O(n^3)$ operations, while for instance `eigs` should take only $O(n^2)$ in typical cases. On the other hand, we point out that in absence of any structure (such as sparsity) in $b(\cdot,\cdot)$, forming the matrix $b(v,\cdot)$ or $b(\cdot,v)$ for a new vector v, an operation which is required at every step in all known iterative algorithms, requires $O(n^3)$ operations. Therefore, the CPU times are somehow indicative of the real complexity of the algorithms, but should be taken with a grain of salt.

The stopping criterion was chosen to be $\|x - a + b(x,x)\| \le n\varepsilon$, with $\varepsilon = 10^{-13}$, for all algorithms.

Table 4.1 shows the results for several choices of λ. The algorithm TH is clearly the slowest, taking far more CPU time than the two competitors. The different behavior of PI when approaching the critical cases is apparent: while the iterations for TH and NM

Table 4.1: CPU time in seconds (and number of iterations in brackets) for TH, NM and PI

λ	TH	NM	PI
0.86	$2.39 \times 10^{+00}$ (11879)	5.09×10^{-03} (14)	4.93×10^{-03} (7)
0.9	6.54×10^{-01} (3005)	4.29×10^{-03} (12)	5.58×10^{-03} (8)
1	2.80×10^{-01} (1149)	3.90×10^{-03} (11)	5.51×10^{-03} (8)
2	9.26×10^{-02} (191)	2.85×10^{-03} (8)	5.51×10^{-03} (8)

(a) E1, several choices of λ

λ	TH	NM	PI
0.5	7.70×10^{-02} (132)	2.33×10^{-03} (8)	5.70×10^{-03} (11)
0.7	7.65×10^{-02} (135)	2.18×10^{-03} (8)	5.61×10^{-03} (11)
0.8	9.36×10^{-02} (313)	2.45×10^{-03} (9)	4.62×10^{-03} (9)
0.84	7.31×10^{-01} (4561)	4.00×10^{-03} (13)	4.11×10^{-03} (8)

(b) E2, several choices of λ

increase, PI seems to be unaffected by the near-singularity of the problem, and in fact the iteration count decreases slightly.

To show further results on the comparison between NM and PI, we report a number of graphs comparing the iteration count and CPU times of the two algorithms. The graphs are not cut at the critical values, but they extend to subcritical cases as well. It is an interesting point to note that when the MBT is subcritical, and thus the minimal solution x^* (extinction probability) is e, the two algorithms have a different behavior: NM (and TH as well) converges to e, while PI skips this solution and converges to a different solution $x > e$. This is because in the derivation of the Perron iteration we chose the solution $\alpha \neq 0$ for the normalization equation, thus explicitly excluding the solution $y = 0$ (i.e., $x = e$).

Figure 4.1 shows a plot of the iteration count of the two methods vs. different values of the parameter λ. While in close-to-critical cases the iteration count for NM has a spike, the one for PI seems to decrease. However, the iteration count comparison is not fair since the steps of the two iterations require a different machine time. Figure 4.2 shows a similar plot, considering the CPU time instead of the iteration count. In order to achieve better accuracy, the plotted times are averages over 100 consecutive runs.

The results now favor the Newton method in most experiments, but in close-to-critical cases the new method achieves better performance. The results are very close to each other, though, so it is to be expected that for larger input sizes or different implementations the differences in the performance of the eigensolver could lead to significant changes in the results.

In order to highlight the performance difference in close-to-critical cases, we report in Figure 4.3 a plot with the CPU times sampled at a larger number of points around the most "interesting" regions of the previous graphs.

The Jacobian (4.12) had spectral radius less than 1 in all the above experiments, a condition which is needed to ensure the convergence of PI. However, this is not true for all possible MBTs. In fact, by setting the parameter λ for E1 to much larger values, we encountered problematic cases in which PI did not converge. Specifically, starting from

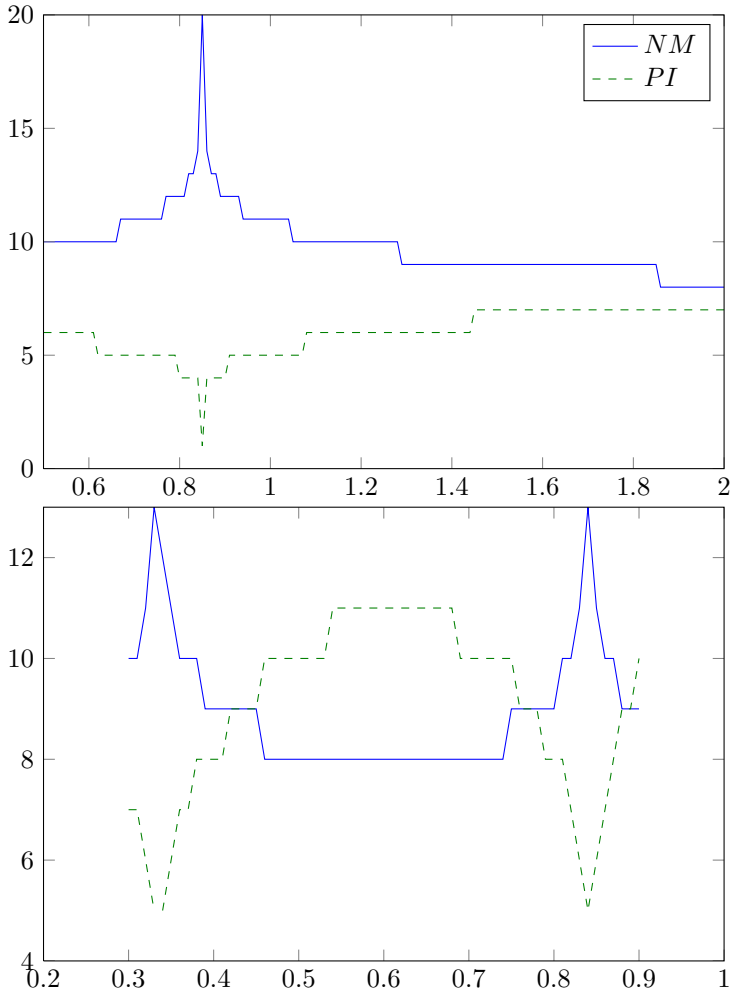


Figure 4.1: Iteration count vs. parameter λ for E1 (top) and E2 (bottom)

$\lambda \approx 78$ the Jacobian (4.12) is larger than 1 and PI does not converge. However, such cases are of little practical interest since they are highly supercritical MBTs, distant from the critical case, and thus they are easily solved with the traditional methods (NM or the customary functional iterations [BKT08]) with a small number of iterations.

The problem E2 is well-posed only for $0 \leq \lambda \leq 1$, otherwise negative entries appear in b, thus the above discussion does not apply.

Along all the experiments reported above, all the matrices H_y appearing in the PI steps always turned out to have positive entries, even in the subcritical problems; thus their Perron vector and values were always well-defined and real.

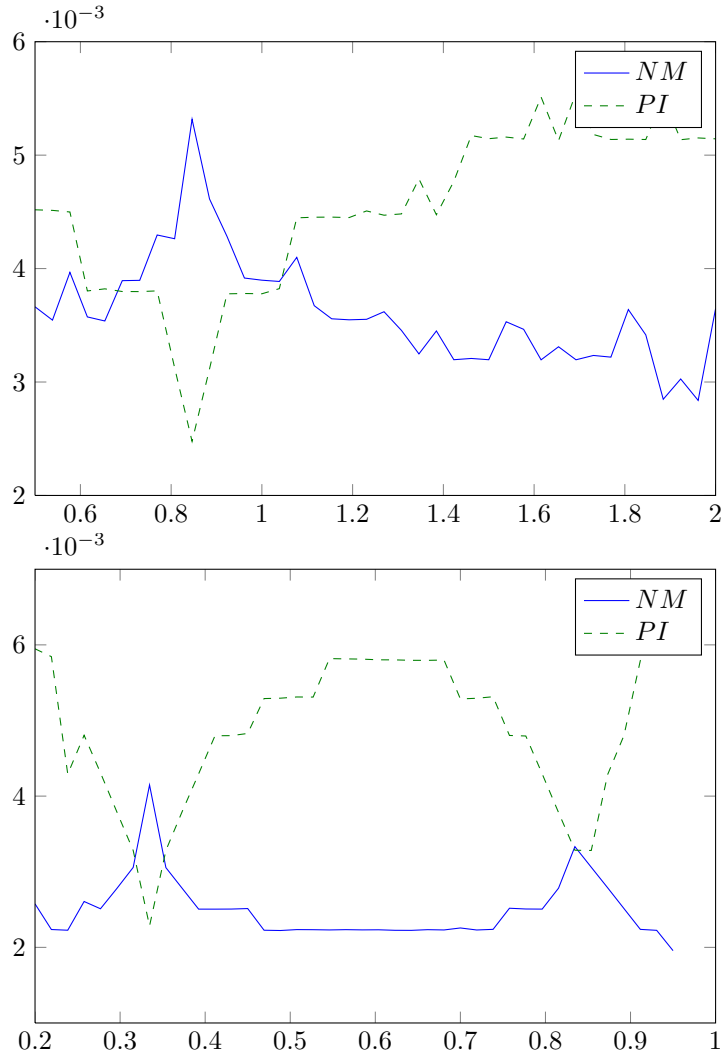


Figure 4.2: CPU time (in sec.) vs. parameter λ for E1 (top) and E2 (bottom)

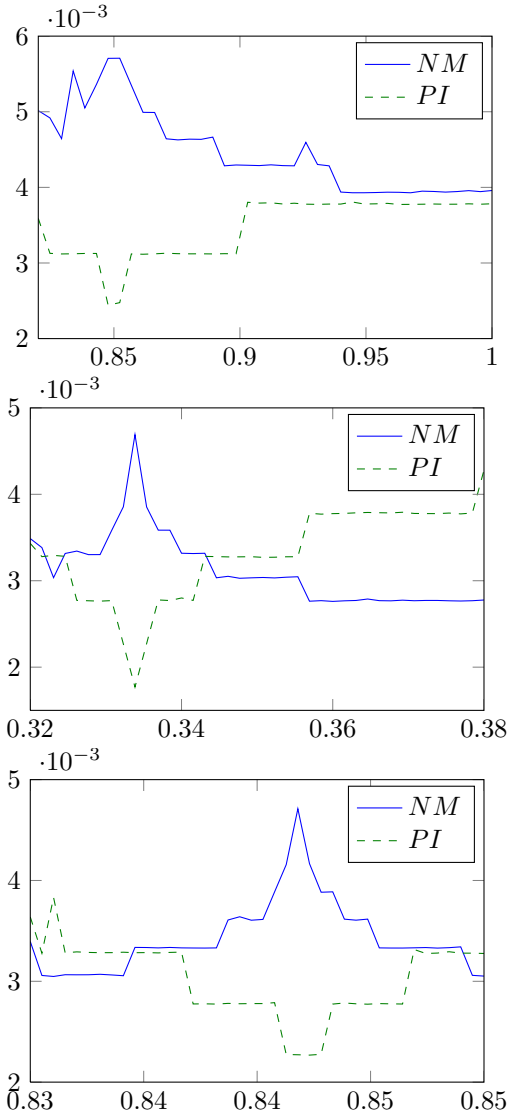


Figure 4.3: Detailed views from Figure 4.2: CPU time (in sec.) vs. parameter λ for E1 (top) and E2 (middle and bottom)

4.7 Conclusions and research lines

We have proposed a new algorithm for solving the quadratic vector equation (1.1) arising in the modeling of Markovian binary trees, based on a Perron iteration [MP10]. The algorithm performs well, both in terms of speed of convergence and accuracy, for close-to-critical problems where the classical methods are slower.

Along the framework that we have exposed, several different choices are possible in the practical implementation of the new algorithm.

One of them is the choice of the bilinear form $b(\cdot,\cdot)$. Equation (4.2) and its solution depend only on the quadratic form $b(t,t)$; however, there are different ways to extend it to a bilinear form $b(s,t)$. This choice ultimately reflects a modeling aspect of the problem: when an individual spawns, it is transformed into two individuals in different states, and we may choose arbitrarily which of them is called the *mother* and which the *child*.

As an example of how this choice affects the solution algorithms, changing the bilinear form may transform the *depth* algorithm into the *order* one and vice versa. The algorithms we proposed depend on the actual choice of the bilinear extension of the quadratic form $b(t,t)$, and the convergence speed is affected by this decision.

A second choice is the normalization of the computed Perron vector: different approaches may be attempted — for instance, minimization of the 1-norm, of the 2-norm, or orthogonality of the residual of (4.3) with respect to a suitably chosen vector — although it is not clear whether we can improve the results of the normalization presented here.

A third choice, crucial in the computational experiments, is the method used to compute the Perron vector. For moderate sizes of the problem, it is cheaper to do a full eigendecomposition of the matrix and extract the eigenvalue with maximum modulus, but for larger problems it pays off to use different specific methods for its computation. All these variants deserve to be better understood, and are now under our investigation.

It would be interesting to see if the assumption that one of the positive solutions to the equation, $(x = e)$ is known can be weakened. In the following chapters, several examples of quadratic vector equations (1.1) appear in which only partial information is known on the solution (e.g., an algebraic Riccati equation (1.5) in which we know that $X^*e = e$). If the Perron vector iteration could be adapted to this case, then we would have a new algorithm for many classes of quadratic equations, with a completely new behavior in the close-to-critical case.

Chapter 5

Unilateral quadratic matrix equations

5.1 Applications

Unilateral quadratic matrix equations appear with positivity assumptions in the context of discrete-time queuing problems [BLM05]. A *QBD process* is a random process on the two-dimensional state space $\mathbb{N} \times \{1, 2, \ldots, n'\}$ for some finite n'. In a state (k, φ), the first coordinate k (called the *level*) represents the number of client in the queue, and the second the state of an outer Markov chain governing the environment space. We assume that transitions are allowed only from level k to the levels $k-1, k, k+1$, i.e., no more than one client can join or leave the queue at any given time, and that the arrival and departure process is independent of the length k of the queue. The (semi-infinite) transition matrix is given by

$$\begin{bmatrix} C+B' & A & & & \\ C & B' & A & & \\ & C & B' & A & \\ & & C & B' & \ddots \\ & & & \ddots & \ddots \end{bmatrix},$$

where $A, B', C \in \mathbb{R}_+^{n' \times n'}$. For this to be a Markov process, $(A + B' + C)e = e$ is assumed. We define $G_{i,j}$ as the probability of first passage from a state $(k+1, i)$ to (k, j), where the latter is the first reached state in level k. Then, one sees that the matrix G is the minimal nonnegative solution to (1.2) and is either substochastic or stochastic.

Using the existence of this solution and other probabilistic techniques, one can prove the following result.

Theorem 5.1 ([BLM05])**.** *If assumptions* (1.3) *hold for equation* (1.2)*, then we are in one of the following cases.*

transient case *the d-stable subspace has dimension n', the d-unstable subspace has dimension $n' - 1$, and there is a simple eigenvalue in 1.*

positive recurrent case *the d-stable subspace has dimension $n' - 1$, the d-unstable subspace has dimension n', and there is a simple eigenvalue in 1.*

null recurrent (critical) case *the d-stable and d-unstable subspace have both dimension $n' - r$, and there are r Jordan chains of length 2 at the r-th roots of unity.*

Moreover, in all cases, the eigenvector relative to 1 is e.

It can be proved that this nomenclature is consistent with the common definitions of transient and recurrent Markov chains. The three cases can be described in probabilistic terms using the so-called *drift* of the associated queuing problem [BLM05].

In all three cases a splitting is present; it is proper in the first two, and partial in the third. In all cases, G spans the canonical semi-stable subspace. In particular, in the positive and null recurrent cases, G has a simple eigenvalue 1.

5.2 Overview of logarithmic and cyclic reduction

Cyclic and logarithmic reduction [BLM05] are two closely related methods for solving (1.2), which have quadratic convergence and a lower computational cost than Newton's method. Both are based on specific properties of the problem, and thus cannot be extended in a straightforward way to other quadratic vector equations.

Logarithmic reduction (LR) [LR93] is based on the fact that if X solves

$$X = AX^2 + C,$$

then it can be shown with algebraic manipulations that $Y = X^2$ solves the equation

$$Y = (I - AC - CA)^{-1}C^2 + (I - AC - CA)^{-1}A^2Y^2, \tag{5.1}$$

with the same structure. Therefore we may start from an approximation $X_1 = C$ to the solution and refine it at each step with a term AY, where Y is (an approximation to) the solution of (5.1). Such an approximation is computed with the same method, and refined successively by applying the same method recursively. The resulting algorithm is reported here as Algorithm 3

Algorithm 3: Logarithmic reduction for E4 [BLM05]

input: A, B, C
output: the minimal solution X to $AX^2 + BX + C = 0$

$A_0 \leftarrow -B^{-1}A$
$C_0 \leftarrow -B^{-1}C$
$X_0 \leftarrow C_0$
$U_0 \leftarrow A_1$
$k \leftarrow 0$
while stopping criterion is not satisfied **do**
  $C_{k+1} \leftarrow (I - A_kC_k - C_kA_k)^{-1}C_k^2$
  $A_{k+1} \leftarrow (I - A_kC_k - C_kA_k)^{-1}A_k^2$
  $X_{k+1} \leftarrow X_k + U_kC_{k+1}$
  $U_{k+1} \leftarrow U_kA_{k+1}$
  $k \leftarrow k + 1$
end while
return X_k

An alternative interpretation of LR [BLM05] arises by defining the matrix-valued function $f : \mathbb{C} \to \mathbb{C}^{n' \times n'}$ as $f(z) = A - z + Cz^2$ and applying the Graeffe iteration $f \mapsto f(z)f(-z)$, which yields a quadratic polynomial in z^2 with the same roots of $f(z)$ plus some additional ones. The derivation and convergence of logarithmic reduction can be derived based on a probabilistic interpretation of the iterates [BLM05].

Cyclic reduction (CR) is a similar algorithm, which is connected to LR by simple algebraic relations (see [BLM05, Theorem 7.5] for more detail). It was originally introduced by Hockney, Buzbee, Golub, Nielson [BGN70, Hoc65] for the Poisson equation over the rectangle, and later adapted to solving matrix equations by Bini and Meini [BM96, BLM05]. Its original derivation is based on performing a block Gaussian elimination on a suitable infinite block tridiagonal matrix [BGN70, BLM05]. We present here essentially the same derivation, but translated in a language that is more suitable to our exposition.

The idea is again trying to build a UQME for $Y = X^2$, but without normalizing the equation before so that the degree-1 term is the identity matrix. We start from the three relations

$$\begin{cases} A + BX + CX^2 = 0, \\ AX + BX^2 + CX^3 = 0, \\ AX^2 + BX^3 + CX^4 = 0, \end{cases} \tag{5.2}$$

which are obtained by multiplying the original equation (1.2) by X and X^2 on the right. We do some manipulations to simplify the terms containing odd powers of X: from the second equation, we subtract the first multiplied by AB^{-1} and the third multiplied by CB^{-1} on the left. Thus we get

$$-AB^{-1}A + (B - AB^{-1}C - CB^{-1}A)Y - CB^{-1}CY^2 = 0,$$

which is again in the same form as (1.2), suitable for recursion. The solution can be obtained by accumulating the current estimates of X, i.e., by approximating in each equation $X \approx -B^{-1}A$ and using a similar estimate of $Y = X^2$ to refine it. However, a different approach is followed usually, based on computing and updating a fourth matrix $\widehat{B}$ according to $\widehat{B} \leftarrow \widehat{B} - CB^{-1}A$ and using it to reconstruct the solution. We report the resulting algorithm as Algorithm 4.

Algorithm 4: Cyclic reduction for E4 [BLM05]

input: A, B, C
output: the minimal solution X to $AX^2 + BX + C = 0$
  $C_0 \leftarrow C$
  $A_0 \leftarrow A$
  $B_0 \leftarrow B$
  $\widehat{B}_0 \leftarrow B$
  $k \leftarrow 0$
  while stopping criterion is not satisfied **do**
    $A_{k+1} \leftarrow -A_k B_k^{-1} A_k$
    $B_{k+1} \leftarrow B_k - A_k B_k^{-1} C_k - C_k B_k^{-1} A_k$
    $\widehat{B}_{k+1} \leftarrow \widehat{B}_k - C_k B_k^{-1} A_k$
    $C_{k+1} \leftarrow -C_k B_k^{-1} C_k$
    $k \leftarrow k + 1$
  end while
  return $-\widehat{B}_k^{-1} A_0$

Notice that both algorithms may break down if the matrices to invert (that is, $(I - A_kC_k - C_kA_k)$ or B_k respectively), are singular.

A proof of the convergence of $-\widehat{B}_k^{-1}A_0$ to the minimal solution X^* can be derived based on arguments similar to those used to derive the CR step, and can be found in [BLM05].

Several results [BM96, BGM02, BLM05] provide necessary and sufficient conditions for the applicability of CR. We report some of the known convergence and applicability results that are needed later on.

Theorem 5.2 ([BLM05]). *If assumptions* (1.3) *hold, then in Algorithm 4,* $A_k, C_k \in \mathbb{R}_+^{n' \times n'}$, *while* $-B_k$ *and* $-\widehat{B}_k$ *are nonsingular M-matrices for each* k. *Thus the algorithm is well-defined.*

Theorem 5.3 ([BLM05]). *Let* $A, B, C \in \mathbb{R}^{n' \times n'}$, *and consider the eigenvalues* $\lambda_1, \ldots, \lambda_{2n'}$ *of the matrix polynomial*

$$s^2 A + sB + C. \tag{5.3}$$

ordered by nondecreasing modulus. Suppose that $\lambda_{n'} \leq 1 \leq \lambda_{n'+1}$, *where at least one of the inequalities is strict (non-null recurrent case). Moreover, suppose that there is a solution* X *with* $\rho(X) \leq \lambda_{n'}$ *and that Algorithm 4 can be carried on with no breakdown.*

Then, $-\widehat{B}_k^{-1} A_0$ *converges to the minimal solution* X^* *quadratically with rate equal to the dominance factor* $\nu = |\lambda_{n'}| \,/\, |\lambda_{n'+1}|$.

Theorem 5.4 ([CCG+09]). *Suppose cyclic reduction (Algorithm 4) is applied to the null recurrent case of* (1.2) *with assumptions* (1.3). *Then,* $-\widehat{B}_k^{-1} A_0$ *converges to the minimal solution* X^* *linearly with rate* $1/2$.

Theorem 5.5 ([BLM05]). *The function* $\varphi(s) = As + B + Cs^{-1}$ *is analytic and invertible for* $|\lambda_{n'}| < |s| < |\lambda_{n'+1}|$. *If in addition there exists a solution to the equation*

$$CX^2 + BX + A = 0 \tag{5.4}$$

with spectral radius $|\lambda_{n'+1}|$, *then the constant coefficient* ψ_0 *of*

$$\psi(s) = \sum_{i=-\infty}^{+\infty} s^i \psi_i = \varphi(s)^{-1} \tag{5.5}$$

is nonsingular and

$$\lim_k B_k = \psi_0^{-1}, \qquad \left\| B_k - \psi_0^{-1} \right\| = O\left(\nu^{2^k}\right).$$

We recall that the assumption on the existence of a solution X^* such that $\rho(X^*) = |\lambda_{n'}|$ is generally satisfied in most applications [BIMP10, BM09]. The computational cost of the CR iteration amounts to 6 matrix products and one LU factorization per step, that is, $\frac{38}{3} n'^3$ ops per step.

Another useful formulation of the cyclic reduction is the functional formulation [BLM05]. Let

$$\varphi_k(s) := s^{-1} C_k + B_k + s A_k,$$

and $\psi_k(s) = \varphi_k(s)^{-1}$, defined for all z for which $\varphi_k(s)$ is nonsingular. Then the CR iteration can be seen as

$$\psi_{k+1}(s^2) = \frac{1}{2}(\psi_k(s) + \psi_k(-s)), \qquad k = 0, 1, \ldots. \tag{5.6}$$

5.3 Generalization attempts

In this section, we attempt to produce algorithms similar to LR and CR for a generic quadratic vector equation. Notice that we cannot look for an equation in X^2 in our vector setting, since x^2 for a vector x has not a clear definition — using e.g. the Hadamard (componentwise) product does not lead to a simple equation. Nevertheless, we may try to find an equation in $b(x,x)$, which is the only quadratic expression that makes sense in our context.

We look for an expression similar to the Graeffe iteration. If x solves $0 = F(x) = Mx - a - b(x,x)$, then it also solves $b(F(x), F(-x)) + b(F(-x), F(x)) = 0$ (notice that a symmetrization is needed), that is,

$$\begin{aligned} b(x - M^{-1}a - M^{-1}b(x,x), x + M^{-1}a + M^{-1}b(x,x)) + \\ b(x + M^{-1}a + M^{-1}b(x,x), x - M^{-1}a - M^{-1}b(x,x)) = 0. \end{aligned}$$

If we set $v_1 = M^{-1}b(x,x)$ and exploit the bilinearity of $b(\cdot,\cdot)$, the above equation reduces to

$$-b(M^{-1}a, M^{-1}a) + \left(M - b(M^{-1}a, \cdot) - b(\cdot, M^{-1}a)\right) v_1 - b(v_1, v_1) = 0, \tag{5.7}$$

which is suitable to applying the same process again. A first approximation to x is given by $M^{-1}a$; if we manage to solve (even approximately) (5.7), this approximation can be refined as $x = M^{-1}a + v_1$. We may apply this process recursively, getting an algorithm similar to logarithmic reduction. The algorithm is reported here as Algorithm 5. It is surprising to

Algorithm 5: A cyclic reduction-like formulation of Newton's method for a quadratic vector equation

input: a, M, b defining a quadratic vector equation (1.1)
output: its minimal solution x^*
  $x \leftarrow 0$, $\tilde{M} \leftarrow M$, $\tilde{a} \leftarrow a$
  while stopping criterion is not satisfied **do**
    $w \leftarrow \tilde{M}^{-1}\tilde{a}$
    $x \leftarrow x + w$
    $\tilde{a} \leftarrow b(w,w)$
    $\tilde{M} \leftarrow \tilde{M} - b(w,\cdot) - b(\cdot,w)$
  end while
  return x

see that this algorithm turns out to be equivalent to Newton's method. In fact, it is easy to prove by induction the following proposition.

Theorem 5.6 ([Pol10b])**.** *Let x_k be the iterates of Newton's method on* (1.1) *starting from $x_0 = 0$. At the kth iteration of the **while** cycle in Algorithm 5, it holds $x = x_k$, $w = x_{k+1} - x_k$, $\tilde{M} = F'_{x_k}$, $\tilde{a} = -F(x_k)$.*

The modified Newton method discussed in Section 3.7 can also be expressed in a form that looks very similar to LR/CR. We may express all the computations of step $k+1$ in terms of $R_{x_k}^{-1}b$ and $R_{x_k}^{-1}a$ only: in fact,

$$R_{x_{k+1}} = R_{x_k} - b(\cdot, x_{k+1} - x_k) = R_{x_k}\left(I - R_{x_k}^{-1}b(\cdot, x_{k+1} - x_k)\right),$$

and thus

$$R_{x_{k+1}}^{-1} R_{x_k} = \left(I - R_{x_k}^{-1} b(\cdot, x_{k+1} - x_k)\right)^{-1}.$$

The resulting algorithm is reported here as Algorithm 6.

Algorithm 6: A cyclic reduction-like formulation of the modified Newton method for a quadratic vector equation

input: a, M, b defining a quadratic vector equation (1.1)
output: its minimal solution x^*
$x \leftarrow 0$, $\tilde{a} \leftarrow a$, $\tilde{b} \leftarrow b$, $\tilde{w} \leftarrow 0$
while stopping criterion is not satisfied **do**
$\tilde{a} \leftarrow (I - \tilde{b}(\cdot, w))^{-1}\tilde{a}$
$\tilde{b} \leftarrow (I - \tilde{b}(\cdot, w))^{-1}\tilde{b}$
$w \leftarrow (I - \tilde{b}(\tilde{a}, \cdot))^{-1}(\tilde{a} - x)$
$x \leftarrow x + w$
end while
return x

The similarities between the two Newton formulations and LR are apparent. In all of them, only two variables (B_{-1} and B_1, $\tilde{a}$ and $\tilde{M}$, $\tilde{a}$ and $\tilde{b}$) are stored and used to carry on the successive steps, and some extra computations and variables are needed to extract the approximation of the solution (X, x) which is refined at each step with a new additive term.

It is a natural question whether there are algebraic relations among LR and Newton methods, or if LR can be interpreted as an inexact Newton method (see e.g. Ortega and Rheinboldt [OR00]), thus providing an alternative proof of its quadratic convergence. However, we were not able to find an explicit relation among the two classes of methods. This is mainly due to the fact that the LR and CR methods are based upon the squaring $X \mapsto X^2$, which we have no means to translate in our vector setting. To this regard we point out that we cannot invert the matrix C, since in many applications it is strongly singular.

Chapter 6

Nonsymmetric algebraic Riccati equations

The research activity concerning the analysis of nonsymmetric algebraic Riccati equations associated with M-matrices and the design of numerical algorithms for their solution has had a strong acceleration in the last decade. Important progresses have been obtained concerning theoretical properties of this class of matrix equations and new effective algorithms relying on the properties of M-matrices have been designed and analyzed [BOT05, BMP10b, BIP08, BILM06, FG06, Guo01, Guo02, Guo06, GH06, GIM07, GL00, HCL05, JL99, LX06, Lu05a, Lu05b].

6.1 Applications

There are two applications where nonsymmetric algebraic Riccati equations (1.5) play an important role: the study of fluid queues models [Rog94, Ram99, Wil82], and the analysis of a neutron transport equation [JL99, Jua01]. In both cases the solution of interest is the minimal nonnegative one.

Application to fluid queues

In the analysis of two dimensional continuous-time Markov processes, called fluid queues, a crucial step is to compute the element-wise minimal nonnegative solution S of the NARE (1.5). In [Asm95, Ram99, Rog94, dSSL02, AR04, BOT05], the fluid flow models are described in terms of a two-dimensional continuous-time Markov process denoted by $\{(X(t), \varphi(t)) : t \geq 0\}$ where $X(t)$ represents the level, while $\varphi(t)$ represents the phase. The phase process $\{\varphi(t) : t \geq 0\}$ is an irreducible Markov chain with space state $\mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{S}_1 = \{1, 2, \ldots, m\}$, $\mathcal{S}_2 = \{m+1, m+2, \ldots, m+n\}$, and infinitesimal generator given by a matrix in the form (1.6). The minimal nonnegative solution $S = (s_{i,j})$ of (1.5) is such that $s_{i,j}$ is the probability that, starting from level x in phase $i \in \mathcal{S}_2$, the process $(X(t), \varphi(t))$ first returns to level x in finite time and does so in phase $j \in \mathcal{S}_1$, while avoiding levels below x. A detailed description of this kind of models can be found in [Ram99].

Application to transport equation

Riccati equations associated with M-matrices also appear in a problem in neutron transport theory, a variation of the one-group neutron transport equation, described in [JL99]. The mathematical model consists in solving an integrodifferential equation; after discretization of this integrodifferential equation, the problem can be expressed as the nonsymmetric algebraic

Riccati equation (1.5) with coefficients (1.10). The resulting $\mathcal{M}$ is a diagonal-plus-rank-1 M-matrix. Due to this additional structure, ad-hoc algorithms can be developed; this is the main subject of Part II.

6.2 Theoretical properties

The properties of (1.5) have been studied, among other papers, in [FG06, Guo01, Guo02, GIM07]; we summarize some of their results. The given results hold for algebraic Riccati equations for which $\mathcal{M}$ is a nonsingular or singular irreducible M-matrix. The case in which $\mathcal{M}$ is singular and reducible is of minor interest.

The dual equation

The *dual equation* of a NARE (1.5) is defined as

$$YBY - YA - DY + C = 0. \tag{6.1}$$

It is the NARE associated with the M-matrix

$$\widetilde{\mathcal{M}} = \begin{bmatrix} A & -B \\ -C & D \end{bmatrix} = \Pi\mathcal{M}\Pi, \qquad \Pi = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$$

which is nonsingular or irreducible singular if and only if $\mathcal{M}$ is so. For the Perron vector $v > 0$ of $\mathcal{M}$ it holds $\mathcal{M}v \geq 0$, thus $\widetilde{\mathcal{M}}\Pi v > 0$ and by Theorem 2.3 this implies that the Z-matrix $\widetilde{\mathcal{M}}$ is an M-matrix.

Existence of nonnegative solutions

The existence of a minimal nonnegative solution for (1.5) follows from the results of Section 3.4; we call this solution S. Observe that the result holds also for the dual equation (6.1), whose minimal solution we denote by T.

The eigenvalue problem associated with the matrix equation

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

The solutions of (1.5) can be described in terms of the invariant subspaces of the matrix $\mathcal{H}$ in (1.8), according to (1.7). Moreover, for each solution X, the eigenvalues $R := D - CX$ are a subset of the eigenvalues of $\mathcal{H}$. Conversely, if the columns of

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}, \qquad \text{with } U_1 \in \mathbb{R}^{n\times n},\ U_2 \in \mathbb{R}^{m\times n}, \tag{6.2}$$

span an invariant subspace of $\mathcal{H}$ and U_1 is nonsingular, then $U_2U_1^{-1}$ is a solution of the Riccati equation [LR95].

Similarly, for the solutions of the dual equation we have

$$\mathcal{H}\begin{bmatrix} Y \\ I_m \end{bmatrix} = \begin{bmatrix} Y \\ I_m \end{bmatrix}(BY - A),$$

and the eigenvalues of $BY - A$ are a subset of those of $\mathcal{H}$.

Theorem 6.1 ([BIMP10])**.** *Let $\mathcal{M}$ be an irreducible M-matrix. Then the eigenvalues of $\mathcal{H}$ have an (m, n) c-splitting. Moreover, the only c-critical eigenvalue that $\mathcal{H}$ may have is a simple 0 eigenvalue.*

Proof. Let $v > 0$ be the Perron vector of $\mathcal{M}$, and let $\lambda \geq 0$ be its Perron value; define $D_v = \operatorname{diag}(v)$. The matrix $\overline{\mathcal{M}} = D_v^{-1}\mathcal{M}D_v$ has the same eigenvalues as $\mathcal{M}$; moreover, it is an M-matrix such that $\overline{\mathcal{M}}e = \lambda e$. Due to the sign structure of M-matrices, this means that $\overline{\mathcal{M}}$ is diagonal dominant (strictly in the nonsingular case). Notice that

$$\overline{\mathcal{H}} = D_v^{-1}\mathcal{H}D_v = \mathcal{K}\overline{\mathcal{M}},$$

thus $\overline{\mathcal{H}}$ is diagonal dominant as well, with m negative and n positive diagonal entries. We apply Gershgorin's theorem [Hog07, Section 14-5] to $\overline{\mathcal{H}}$; due to the diagonal dominance, the Gershgorin circles never cross the imaginary axis (in the singular case, they are tangent in 0). Thus, by using a continuity argument we can say that m eigenvalues of $\overline{\mathcal{H}}$ lie in the negative half-plane and n in the positive one, and the only eigenvalues on the imaginary axis are the zero ones. But since $\mathcal{H}$ and $\overline{\mathcal{H}}$ are similar, they have the same eigenvalues.

If $\mathcal{H} = \mathcal{K}\mathcal{M}$ has a zero eigenvalue with right eigenvector u, then u is an eigenvector for $\mathcal{M}$ as well. It must be the Perron vector, with eigenvalue 0. Therefore it is simple. □

We can give a more precise result on the location of the eigenvalues of H, after defining the *drift* of the Riccati equation [BOT05]. When $\mathcal{M}$ is a singular irreducible M-matrix, by the Perron–Frobenius theorem, the eigenvalue 0 is simple, there are positive vectors u and v such that

$$u^T\mathcal{M} = 0, \qquad \mathcal{M}v = 0, \tag{6.3}$$

and both the vectors u and v are unique up to a scalar factor.

In fluid queues problems, v coincides with the vector of ones. In general v and u can be computed by performing the LU factorization of the matrix $\mathcal{M}$, say $\mathcal{M} = LU$, and solving the two triangular linear systems $u^TL = [0, \ldots, 0, 1]$ and $Uv = 0$ (see [Hog07, Sec. 54-12]).

Using the technique of Theorem 6.1, we may apply a positive diagonal scaling to $\mathcal{M}$ to impose that the right eigenvalue v is e; therefore, in the theoretical analysis we may assume without loss of generality $v = e$ when needed.

We consider the quantity

$$\mu = u_2^Tv_2 - u_1^Tv_1 = -u^T\mathcal{K}v, \quad \text{with } u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},\ v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \tag{6.4}$$

where $u_1, v_1 \in \mathbb{R}^n$ and $u_2, v_2 \in \mathbb{R}^m$. The number μ is called *drift* and determines some spectral properties of the Riccati equation, in a result similar to Theorem 5.1.

Theorem 6.2 ([Guo01, Guo06])**.** *If $\mathcal{M}$ is a nonsingular M-matrix, then we are in the following case.*

nonsingular case *the c-stable space has dimension m and the c-unstable subspace has dimension n, and there are no c-critical eigenvalues.*

If $\mathcal{M}$ is a singular irreducible M-matrix, then we are in one of the following cases.

transient case *if $\mu > 0$, the c-stable subspace has dimension m, the c-unstable subspace has dimension $n - 1$, and there is a simple eigenvalue in 0.*

positive recurrent case *if $\mu < 0$, the c-stable subspace has dimension $m-1$, the d-unstable subspace has dimension n, and there is a simple eigenvalue in 0.*

null recurrent (critical) case *if $\mu = 0$, the c-stable space has dimension $m - r$, the c-unstable subspace has dimension $n - r$, and there are r Jordan chains of length 2 at the r-th roots of unity.*

The *close to null recurrent* case, i.e., the case $\mu \approx 0$, deserves a particular attention, since it corresponds to an ill-conditioned null eigenvalue for the matrix $\mathcal{H}$. In fact, if u and v are normalized such that $\|u\|_2 = \|v\|_2 = 1$, then $1/|\mu|$ is the condition number of the null eigenvalue for the matrix $\mathcal{H}$ (see [GVL96]).

When $\mathcal{M}$ is singular irreducible, for the Perron–Frobenius Theorem the eigenvalue 0 is simple, therefore $\mathcal{H}$ has a one dimensional kernel, too, and $u^T\mathcal{K}$ and v are the unique (up to a scalar constant) left and right eigenvectors, respectively, corresponding to the eigenvalue 0. However the algebraic multiplicity of 0 as an eigenvalue of $\mathcal{H}$ can be 2; in that case, the Jordan form of $\mathcal{H}$ has a 2×2 Jordan block corresponding to the 0 eigenvalue and it holds $u^T\mathcal{K}v = 0$ [HSC06].

The relationship between the eigenvalues of $\mathcal{H}$ and the minimal solution is precised in the following theorem.

Theorem 6.3 ([Guo01, Guo06])**.** *Let $\mathcal{M}$ be a nonsingular or a singular irreducible M-matrix, and let $\lambda_1, \ldots, \lambda_{m+n}$ be the eigenvalues of $\mathcal{H}$ ordered by nonincreasing real part. Then,*

1. *the eigenvalues λ_n and λ_{n+1} are real and*

$$\Re(\lambda_{n+m}) \leq \cdots \leq \Re(\lambda_{n+2}) < \lambda_{n+1} \leq 0 \leq \lambda_n < \Re(\lambda_{n-1}) \leq \cdots \leq \Re(\lambda_1). \tag{6.5}$$

2. *The minimal nonnegative solutions S of (1.5) and T of the dual equation (6.1) are such that $D - CS$, $A - SC$, $A - BT$, $D - TB$ are M-matrices*

3. *$\sigma(D-CS) = \sigma(D-TB) = \{\lambda_1, \ldots, \lambda_n\}$ and $\sigma(A-SC) = \sigma(A-BT) = \{-\lambda_{n+1}, \ldots, -\lambda_{n+m}\}$.*

4. *If the equation is*

   - *nonsingular, then $\lambda_{n+1} < 0 < \lambda_n$.*
   - *transient, then $\lambda_n > 0$ and $\lambda_{n+1} = 0$.*
   - *positive recurrent, then $\lambda_n = 0$ and $\lambda_{n+1} < 0$;*
   - *null recurrent, then $\lambda_n = \lambda_{n+1} = 0$ and there is a Jordan chain of length 2 relative to the eigenvalue 0;*

We call λ_n and λ_{n+1} the *central eigenvalues* of $\mathcal{H}$. If $\mathcal{H}$ (and thus $\mathcal{M}$) is nonsingular, then the central eigenvalues lie on two different half planes, so the splitting is proper. In this case, $D - CS$ and $A - SC$ are c-unstable matrices, thus in view of (1.7) the matrices

$$\begin{bmatrix} I_n \\ S \end{bmatrix} \qquad\qquad \begin{bmatrix} T \\ I_m \end{bmatrix} \tag{6.6}$$

span respectively the unstable and stable subspaces of $\mathcal{H}$. By comparing Theorem 6.3 with our definition of canonical semi-stable and semi-unstable subspaces, we see that in the case in which $\mathcal{M}$ is singular, they span respectively the canonical semi-stable and semi-unstable subspaces.

The next result, presented in [GIM07], shows how we can reduce the case $\mu < 0$ to the case $\mu > 0$ and conversely. This property enables us to restrict our interest only to the case $\mu \leq 0$.

Lemma 6.4 ([GIM07])**.** *The matrix S is the minimal nonnegative solution of* (1.5) *if and only if $Z = S^T$ is the minimal nonnegative solution of the equation*

$$XC^TX - XA^T - D^TX + B^T = 0. \tag{6.7}$$

When $\mathcal{M}$ is singular and irreducible, (1.5) *is transient if and only if* (6.7) *is positive recurrent.*

Proof. The first part is easily shown by transposing both sides of (1.5). The M-matrix corresponding to (6.7) is

$$\mathcal{M}_t = \begin{bmatrix} A^T & -C^T \\ -B^T & D^T \end{bmatrix}.$$

Since the left and right Perron vectors of $\mathcal{M}_t$ are given by

$$\begin{bmatrix} v_2^T & v_1^T \end{bmatrix} \mathcal{M}_t = 0, \qquad \mathcal{M}_t \begin{bmatrix} u_2 \\ u_1 \end{bmatrix} = 0,$$

the second part follows. □

Theorem 6.5 ([Guo01, Guo02, GH06])**.** *Let $\mathcal{M}$ be singular and irreducible, and let S and T be the minimal nonnegative solutions of* (1.5) *and* (6.1)*, respectively. Then the following properties hold:*

(a) if $\mu < 0$, then $Sv_1 = v_2$ and $Tv_2 < v_1$;

(b) if $\mu = 0$, then $Sv_1 = v_2$ and $Tv_2 = v_1$;

(c) if $\mu > 0$, then $Sv_1 < v_2$ and $Tv_2 = v_1$.

Remark 6.6. When $\mu \geq 0$, from Lemma 6.4 and Theorem 6.5 we deduce that the minimal nonnegative solution S of (1.5) is such that $u_2^T S = u_1^T$.

The Fréchet derivative of the Riccati operator

We define the Riccati operator $\mathfrak{R}(X)$ and the operator associated with the dual equation (6.1) as

$$\begin{aligned} \mathfrak{R}(X) &:= XCX - AX - XD + B, \\ \mathfrak{D}(Y) &:= YBY - DY - YA + C, \end{aligned} \tag{6.8}$$

where X and Y^T are $m \times n$ matrices.

The Fréchet derivative $\mathfrak{R}'$ is given by

$$\mathfrak{R}'_X(W) = WCX + XCW - AW - WD. \tag{6.9}$$

The Fréchet derivative $W \to \mathfrak{R}'_X(W)$ is a linear operator which can be represented by the matrix

$$\Delta_X = (CX - D)^T \otimes I_m + I_n \otimes (XC - A). \tag{6.10}$$

We say that a solution X of the matrix equation (1.5) is critical if the matrix Δ_X is singular.

From the properties of Kronecker product [Hog07, Sec. 10.4], it follows that the eigenvalues of Δ_X are the sums of those of $CX - D$ and $XC - A$. If $X = S$, where S is the minimal nonnegative solution, then $D - CS$ and $A - SC$ are M-matrices by Theorem 6.3, and thus all the eigenvalues of Δ_S have nonpositive real parts. Moreover, since $D - CS$ and

$A - SC$ are M-matrices then $-\Delta_S$ is an M-matrix. The minimal nonnegative solution S is critical if and only if both M-matrices $D - CS$ and $A - SC$ are singular, thus only in the null recurrent case.

Moreover, if $0 \leq X \leq S$ then $D - CX \geqslant D - CS$ and $A - XC \geq A - SC$ are nonsingular M-matrices by Lemma 2.3, thus $-\Delta_X$ is a nonsingular M-matrix.

The number of positive solutions

We know from Section 3.4 that there is a minimal nonnegative solution S of the NARE.

In [FG06], it is shown that if M is nonsingular or singular irreducible with $\mu \neq 0$, then there exists a second solution S_+ such that $S_+ > S$ and S_+ is obtained by a rank one correction of the matrix S. More precisely, the following result holds [FG06].

Theorem 6.7 ([FG06])**.** *If $\mathcal{M}$ is irreducible nonsingular or irreducible singular with $\mu \neq 0$, then there exists a second positive solution S_+ of* (1.5) *given by*

$$S_+ = S + kab^T,$$

where $k = (\lambda_n - \lambda_{n+1})/b^T Ca$, a is such that $(A - SC)a = -\lambda_{n+1}a$ and b is such that $b^T(D - CS) = \lambda_n b^T$.

We prove that there are exactly two nonnegative solutions in the noncritical case and only one in the critical case. In order to prove this result it is useful to study the form of the Jordan chains of an invariant subspace of $\mathcal{H}$ corresponding to a positive solution.

Lemma 6.8. *Let $\mathcal{M}$ be irreducible and let Σ be any positive solution of* (1.5)*. Denote by $\eta_1, \ldots, \eta_n$ the eigenvalues of $D - C\Sigma$ ordered by nondecreasing real part. Then η_1 is real, and there exists a positive eigenvector v of $\mathcal{H}$ associated with η_1. Moreover, any other vector independent of v, belonging to Jordan chains of $\mathcal{H}$ corresponding to $\eta_1, \ldots, \eta_n$ cannot be positive or negative.*

Proof. Since Σ is a solution of (1.5), then one has

$$\mathcal{H} \begin{bmatrix} I \\ \Sigma \end{bmatrix} = \begin{bmatrix} I \\ \Sigma \end{bmatrix} (D - C\Sigma).$$

Since $D - CS$ is an irreducible M-matrix by Theorem 6.3, and $\Sigma \geq S$ (S is the minimal positive solution), then $D - C\Sigma$ is an irreducible Z-matrix and thus can be written as $sI - N$ with N nonnegative and irreducible. Then by Theorem 2.1 and Corollary 2.2 η_1 is a simple real eigenvalue of $D - C\Sigma$, the corresponding eigenvector can be chosen positive and there are no other positive or negative eigenvectors or Jordan chains corresponding to any of the eigenvalues. Let $P^{-1}(D - C\Sigma)P = K$ be the Jordan canonical form of $D - C\Sigma$, where the first column of P is the positive eigenvector corresponding to η_1. Then we have

$$\mathcal{H} \begin{bmatrix} P \\ \Sigma P \end{bmatrix} = \begin{bmatrix} P \\ \Sigma P \end{bmatrix} K.$$

Thus, the columns of $\begin{bmatrix} P \\ \Sigma P \end{bmatrix}$ are the Jordan chains of $\mathcal{H}$ corresponding to $\eta_1, \ldots, \eta_n$, and there are no positive or negative columns, except for the first one. □

Theorem 6.9 ([BIMP10])**.** *Let $\mathcal{M}$ be irreducible. Equation* (1.5) *has a unique positive solution in the null recurrent case, and exactly two positive solutions otherwise.*

Proof. From Lemma 6.8 applied to S it follows that $\mathcal{H}$ has a positive eigenvector corresponding to λ_n, and no other positive or negative eigenvectors or Jordan chains corresponding to $\lambda_1, \ldots, \lambda_n$. Let T be the minimal nonnegative solution of the dual equation (6.1). Then

$$\mathcal{H} \begin{bmatrix} T \\ I \end{bmatrix} = \begin{bmatrix} T \\ I \end{bmatrix} (-(A - BT)).$$

As in the proof of Lemma 6.8, we can prove that $\mathcal{H}$ has a positive eigenvector corresponding to the eigenvalue λ_{n+1} and no other positive or negative eigenvectors or Jordan chains corresponding to $\lambda_{n+1}, \ldots, \lambda_{n+m}$.

If the equation is not null recurrent, then $\lambda_n > \lambda_{n+1}$, and there are only two linearly independent positive eigenvectors corresponding to real eigenvalues. By Lemma 6.8, there can be at most two solutions corresponding to $\lambda_n, \lambda_{n-1}, \ldots, \lambda_1$, and to $\lambda_{n+1}, \lambda_{n-1}, \ldots, \lambda_1$, respectively. Since it is know from Theorem 6.7 that there exist at least two positive solutions, thus (1.5) has exactly two positive solutions.

If the NARE is null recurrent, there is only one positive eigenvector corresponding to $\lambda_n = \lambda_{n+1}$, and the unique solution of (1.5) is obtained by the Jordan chains corresponding to $\lambda_n, \lambda_{n-1}, \ldots, \lambda_1$. □

Perturbation analysis for the minimal solution

We conclude this section with a result of Guo and Higham [GH06] who perform a qualitative description of the perturbation of the minimal nonnegative solution S of a NARE (1.5) associated with an M-matrix.

The result is split in two theorems where an M-matrix $\widetilde{\mathcal{M}}$ is considered which is obtained by means of a perturbation of M. Here, we denote by $\widetilde{S}$ the minimal nonnegative solution of the perturbed Riccati equation associated with $\widetilde{\mathcal{M}}$.

Theorem 6.10 ([GH07])**.** *If* (1.5) *is not null recurrent, then there exist constants* $\gamma > 0$ *and* $\varepsilon > 0$ *such that* $\|\widetilde{S} - S\| \leq \gamma \|\widetilde{\mathcal{M}} - \mathcal{M}\|$ *for all* $\widetilde{\mathcal{M}}$ *with* $\|\widetilde{\mathcal{M}} - \mathcal{M}\| < \varepsilon$.

Theorem 6.11 ([GH07])**.** *If* (1.5) *is null recurrent, then there exist constants* $\gamma > 0$ *and* $\varepsilon > 0$ *such that*

1. $\|\widetilde{S} - S\| \leq \gamma \|\widetilde{\mathcal{M}} - \mathcal{M}\|^{1/2}$ *for all* $\widetilde{\mathcal{M}}$ *with* $\|\widetilde{\mathcal{M}} - \mathcal{M}\| < \varepsilon$;
2. $\|\widetilde{S} - S\| \leq \gamma \|\widetilde{\mathcal{M}} - \mathcal{M}\|$ *for all singular* $\widetilde{\mathcal{M}}$ *with* $\|\widetilde{\mathcal{M}} - \mathcal{M}\| < \varepsilon$.

Therefore, when $\mu = 0$ or if $\mu \approx 0$, one should expect poor numerical performance even if the algorithm used for approximating S is backward stable. The rounding errors introduced to represent the input values of $\mathcal{M}$ in the floating point representation with precision ϵ may generate an error of the order $\sqrt{\epsilon}$ in the solution S. A solution to this problem for the null recurrent case is described in [GIM07].

6.3 Schur method

In the following, we give a brief introduction to several numerical methods for computing the minimal nonnegative solution of a NARE (1.5) associated with an M-matrix. Other numerical algorithms are introduced later in Chapter 7.

Here we consider the case where $\mathcal{M}$ is nonsingular or is singular, irreducible and $\mu \leq 0$. The case $\mu > 0$ can be reduced to the case $\mu < 0$ by means of Lemma 6.4.

A classical approach for solving equation (1.5) is to use the (ordered) Schur decomposition of the matrix $\mathcal{H}$ to compute the invariant subspace corresponding to the minimal solution S. This approach for the symmetric algebraic Riccati equation was first presented by Laub in 1979 [Lau79]. Concerning the NARE, a study of this method in the singular and critical case was done by Guo [Guo06] who presented a modified Schur method for the critical or near critical case ($\mu \approx 0$).

As explained in Section 6.2, finding the minimal solution S of (1.5) is equivalent to finding a basis of the c-semi-unstable invariant subspace of $\mathcal{H}$ relative to the eigenvalues of $D - CS$.

A method for finding an invariant subspace is obtained by computing a semi-ordered Schur form of $\mathcal{H}$, that is, computing an orthogonal matrix Q and a quasi upper-triangular matrix T such that $Q^*\mathcal{H}Q = T$, where T is block upper triangular with diagonal blocks $T_{i,i}$ of size at most 2. The semi-ordering means that if $T_{i,i}$, $T_{j,j}$ and $T_{k,k}$ are diagonal blocks having eigenvalues with positive, null and negative real parts, respectively, then $i < j < k$.

A semi-ordered Schur form can be computed in two steps:

- Compute a real Schur form of $\mathcal{H}$ by the customary Hessenberg reduction followed by the application of the QR algorithm as described in [GVL96].

- Swap the diagonal blocks by means of orthogonal transformations as described in [BD93].

The minimal solution of the NARE can then be obtained from the first n columns of the matrix Q: if we partition them as in (6.2), then $S = U_2U_1^{-1}$.

In the critical case this method does not work, since there is no way to choose an invariant subspace relative to the first n eigenvalues; moreover in the near critical case where $\mu \approx 0$, there is lack of accuracy since the 0 eigenvalue is ill-conditioned. However, the modified Schur method given by C.-H. Guo [GH06] overcomes these problems.

The cost of this algorithm is $200n^3$ ops for computing the Schur form, plus the cost for the reordering, which may vary depending on how many blocks need to be reordered in the Schur form [Guo06].

6.4 Functional iterations and Newton's method

Functional iterations and Newton's method may be performed according to the framework introduced in Chapter 3.

Fixed-point iterations may be implemented based on suitable splittings of A and D, that is $A = A_1 - A_2$ and $D = D_1 - D_2$, with A_1, D_1 chosen to be M-matrices and $A_2, D_2 \geq 0$. The form of the iterations is then

$$A_1X_{k+1} + X_{k+1}D_1 = X_kCX_k + X_kD_2 + A_2X_k + B, \tag{6.11}$$

and at each step a Sylvester equation of the form $M_1X + XM_2 = N$ must be solved.

Some possible choices for the splitting are:

1. A_1 and D_1 are the diagonal parts of A and D, respectively;

2. A_1 is the lower triangular part of A and D_1 the upper triangular part of D;

3. $A_1 = A$ and $D_1 = D$.

The solution X_{k+1} of the Sylvester equation can be computed, for instance, by using the Bartels and Stewart method [BS72], as in Matlab's `sylvsol` function of the Nick Higham Matrix Function toolbox [Higb].

The cost of this computation is about $60n^3$ ops including the computation of the Schur form of the coefficients A_1 and D_1 [Hig08].

However, observe that for the first splitting, A_1 and D_1 are diagonal matrices and the Sylvester equation can be solved with $O(n^2)$ ops; for the second splitting, the matrices A_1 and D_1 are already in Schur form. This substantially reduces the cost of the application of the Bartels and Stewart method to $2n^3$. Concerning the third iteration, observe that the matrix coefficients A_1 and D_1 are independent of the iteration. Therefore, the computation of their Schur form must be performed only once.

We have also an asymptotic convergence result.

Theorem 6.12 ([Guo01])**.** *For the fixed-point iterations* (6.11) *with* $X_0 = 0$*, it holds that*

$$\limsup \sqrt[k]{\|X_k - S\|} = \rho((I \otimes A_1 + D_1^T \otimes I)^{-1}(I \otimes (A_2 + SC) + (D_2 + CS)^T \otimes I).$$

These iterations have linear convergence which turns to sublinear in the critical case. The computational cost varies from $8n^3$ arithmetic operations per step for the first splitting, to $64n^3$ for the first step plus $10n^3$ for each subsequent step for the last splitting. The most expensive iteration is the third one which, on the other hand, has the highest (linear) convergence speed.

Newton's iteration was first applied to the symmetric algebraic Riccati equation by Kleinman in 1968 [Kle68] and later on by various authors. In particular, Benner and Byers [BB98] complemented the method with an optimization technique (exact line search) 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 [GL00], and a nice convergence result was given by Guo and Higham in [GH06].

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 [GL00]. 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.

The Newton method for a NARE [GL00] consists in the iteration (3.6) which, in view of (6.9), can be written explicitly as

$$(A - X_kC)X_{k+1} + X_{k+1}(D - CX_k) = B - X_kCX_k. \tag{6.12}$$

Therefore, the matrix X_{k+1} is obtained by solving a Sylvester equation. This linear equation is defined by the matrix

$$\Delta_{X_k} = (D - CX_k)^T \otimes I_m + I_n \otimes (A - X_kC)$$

which is a nonsingular M-matrix if $0 \leq X_k < S$, as shown in section 6.2.

In the noncritical case, $\mathfrak{R}'_S$ is nonsingular, and the iteration is quadratically convergent in a neighborhood of the minimal nonnegative solution S by the traditional results on Newton's method (see e.g. [Kan52]). Moreover, the convergence is monotonic, according to Theorem 3.9.

In [GL00], a hybrid method was suggested, which consists in performing a certain number of iterations of a linearly convergent algorithm, such as the ones of Section 6.4, and then using the computed value as the starting point for Newton's method.

At each step of Newton's iteration, the largest computational work is given by the solution of the Sylvester equation (6.12). The overall cost of Newton's iteration is $66n^3$ ops.

It is worth noting that in the critical and near critical cases, the matrix Δ_k becomes almost singular as X_k approaches the solution S; therefore, some numerical instability is to be expected. Such instability can be removed by means of a suitable technique [GIM07, BIMP10].

6.5 Matrix sign method

In the case of a nonsingular equation, the solution can be computed using the matrix sign function. The matrix sign function [Hig08] is defined for all matrices without purely imaginary eigenvalues as the matrix extension of the sign function

$$\operatorname{sgn} x = \begin{cases} +1 & \text{if } \Re(x) > 0, \\ -1 & \text{if } \Re(x) < 0. \end{cases}$$

Alternatively, if the Jordan form of $\mathcal{H}$ is ordered such that

$$\mathcal{H} = U \begin{bmatrix} J_s & 0 \\ 0 & J_u \end{bmatrix} U^H,$$

where J_s is c-stable and J_u is c-unstable, then the matrix sign function is given by

$$\operatorname{sgn} \mathcal{H} := U \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} U^H.$$

It follows from this definition that the unstable space is given by $\ker \operatorname{sgn}(\mathcal{H}) - I$, thus if we compute

$$\ker \operatorname{sgn}(\mathcal{H}) - I = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix},$$

then the kernel is n-dimensional, U_1 is nonsingular and $S = U_2 U_1^{-1}$.

Popular choices for computing the matrix sign function include the Schur method, which leads essentially to the same algorithm as the one in Section 6.3, the Newton iteration and the Padé family of iterations. We describe here the Newton iteration, and refer the reader to [Hig08] for more detail. The Newton iteration for the matrix sign (NMS iteration), which should not be confused with the Newton method, is the iteration

$$\mathcal{H}_{k+1} = \frac{\gamma_k}{2}(\mathcal{H}_k + \mathcal{H}_k^{-1}), \qquad \mathcal{H}_0 = \mathcal{H} \tag{6.13}$$

where $\gamma_k \in \mathbb{R}$ is a suitable scaling factor. It converges quadratically to the sign function of $\mathcal{H}$, and the number of iterations can be bounded in terms of the condition number of the matrix $\mathcal{H}$. Unfortunately, when the matrix is close to null recurrent this condition number is expected to be large, due to the presence of an eigenvalue close to zero. Therefore, the NMS iteration is not recommended for the robust solution of NAREs. However, it is interesting to point out its relationship with the algorithms that are introduced in the following sections.

6.6 Block swapping in pencil arithmetic

Pencil arithmetic [BB06] is a technique to extend several common operations (matrix sums, matrix products) to matrix pencils. From a practical point of view, it provides a method to implement the linear algebra basic operations on matrices of the form $E^{-1}A$ while storing

and operating on the pair (E, A) only. This allows extending such operations in a numerically stable fashion even when E is singular or ill-conditioned.

A basic kernel that is needed in several pencil arithmetic computations is the following (*CS-AB problem* [SQO04]): given two matrices $\mathcal{E}, \mathcal{A} \in \mathbb{R}^{k\times k}$, compute $\widetilde{\mathcal{E}}, \widetilde{\mathcal{A}} \in \mathbb{R}^{k\times k}$ such that

$$\widetilde{\mathcal{A}}\mathcal{E} = \widetilde{\mathcal{E}}\mathcal{A}, \text{ or } \begin{bmatrix}\widetilde{\mathcal{E}} & \widetilde{\mathcal{A}}\end{bmatrix}\begin{bmatrix}\mathcal{A}\\ -\mathcal{E}\end{bmatrix} = 0, \tag{6.14}$$

and rank $\begin{bmatrix}\widetilde{\mathcal{E}} & \widetilde{\mathcal{A}}\end{bmatrix} = k$.

Clearly several choices are possible: if the pair $(\widetilde{\mathcal{E}}, \widetilde{\mathcal{A}})$ is a solution, then so is $(S\widetilde{\mathcal{E}}, S\widetilde{\mathcal{A}})$ for any nonsingular $S \in \mathbb{R}^{k\times k}$.

A simple approach, which fails if $\mathcal{E}$ is singular or ill-conditioned, is $(\widetilde{\mathcal{E}}, \widetilde{\mathcal{A}}) = (S\mathcal{A}\mathcal{E}^{-1}, S)$, for any nonsingular S. A more robust but computationally expensive choice is computing the QR decomposition

$$\begin{bmatrix}\mathcal{A}\\ -\mathcal{E}\end{bmatrix} = \begin{bmatrix}Q_1 & Q_2\\ Q_3 & Q_4\end{bmatrix}\begin{bmatrix}R\\ 0\end{bmatrix}$$

and taking $(\widetilde{\mathcal{E}}, \widetilde{\mathcal{A}}) = (Q_2^T, Q_4^T)$.

We propose a third approach, which is useful to preserve a special block structure that is introduced later.

Theorem 6.13 ([MP])**.** *Let $\mathcal{A}, \mathcal{E} \in \mathbb{R}^{k\times k}$ be given, $M \in \mathbb{R}^{2k\times 2k}$ be a nonsingular matrix, and let*

$$M\begin{bmatrix}\mathcal{A}\\ -\mathcal{E}\end{bmatrix} = \begin{bmatrix}X\\ Y\end{bmatrix}$$

If X is nonsingular, then for any nonsingular $S \in \mathbb{R}^{k\times k}$

$$\begin{bmatrix}-SYX^{-1} & S\end{bmatrix} M^{-1}$$

is a solution to the CS-AB problem.

Proof. Direct computation. □

Notice that the simple solution corresponds to $S = I$ and

$$M = \begin{bmatrix}0 & I_k\\ I_k & 0\end{bmatrix},$$

while the more robust approach corresponds to $S = I$ and $M = Q^T$, with Q the orthogonal factor of the QR factorization.

If $k = n + m$, we may divide the matrices into blocks of sizes $(n, m) \times (n, m)$

$$\mathcal{A} = \begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}, \quad \mathcal{E} = \begin{bmatrix}E_{11} & E_{12}\\ E_{21} & E_{22}\end{bmatrix}, \quad \widetilde{\mathcal{A}} = \begin{bmatrix}\widetilde{A}_{11} & \widetilde{A}_{12}\\ \widetilde{A}_{21} & \widetilde{A}_{22}\end{bmatrix}, \quad \widetilde{\mathcal{E}} = \begin{bmatrix}\widetilde{E}_{11} & \widetilde{E}_{12}\\ \widetilde{E}_{21} & \widetilde{E}_{22}\end{bmatrix}, \tag{6.15}$$

Choosing now

$$M = \begin{bmatrix}0 & 0 & I_n & 0\\ 0 & I_m & 0 & 0\\ I_n & 0 & 0 & 0\\ 0 & 0 & 0 & I_m\end{bmatrix}$$

transforms the original CS-AB problem into one of the form

$$\left[\begin{array}{cc|cc} \widetilde{A}_{11} & \widetilde{E}_{12} & \widetilde{E}_{11} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{E}_{22} & \widetilde{E}_{21} & \widetilde{A}_{22} \end{array}\right] \left[\begin{array}{cc} -E_{11} & -E_{12} \\ A_{21} & A_{22} \\ \hline A_{11} & A_{12} \\ -E_{21} & -E_{22} \end{array}\right] = 0$$

or

$$\begin{bmatrix} \widetilde{A}_{11} & \widetilde{E}_{12} \\ \widetilde{A}_{21} & \widetilde{E}_{22} \end{bmatrix} \begin{bmatrix} -E_{11} & -E_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} \widetilde{E}_{11} & \widetilde{A}_{12} \\ \widetilde{E}_{21} & \widetilde{A}_{22} \end{bmatrix} \begin{bmatrix} -A_{11} & -A_{12} \\ E_{21} & E_{22} \end{bmatrix}.$$

The following result can then be proved from Theorem 6.13 or by direct inspection from the latter equality.

Theorem 6.14 ([MP])**.** *Let the matrices $\mathcal{A}$, $\mathcal{E}$ be partitioned as in* (6.15)*, and the blocks*

$$\begin{bmatrix} \widetilde{E}_{11} & \widetilde{A}_{12} \\ \widetilde{E}_{21} & \widetilde{A}_{22} \end{bmatrix}$$

be given. If

$$\begin{bmatrix} -E_{11} & -E_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

is nonsingular, then $\widetilde{\mathcal{A}}$, $\widetilde{\mathcal{E}}$ can be completed to a solution to the CS-AB problem by setting

$$\begin{bmatrix} \widetilde{A}_{11} & \widetilde{E}_{12} \\ \widetilde{A}_{21} & \widetilde{E}_{22} \end{bmatrix} = \begin{bmatrix} \widetilde{E}_{11} & \widetilde{A}_{12} \\ \widetilde{E}_{21} & \widetilde{A}_{22} \end{bmatrix} \begin{bmatrix} -A_{11} & -A_{12} \\ E_{21} & E_{22} \end{bmatrix} \begin{bmatrix} -E_{11} & -E_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1}.$$

The CS-AB solution described in the latter theorem has the same computational cost as the simple one $(\widetilde{\mathcal{E}}, \widetilde{\mathcal{A}}) = (S\mathcal{A}\mathcal{E}^{-1}, S)$, but requires the inversion of a different matrix. Thus when $\mathcal{E}$ is singular or ill-conditioned, it may be advisable to use this solution instead.

6.7 Inverse-free NMS iteration

As an example of the use of pencil arithmetic and of its benefits, we outline an inverse-free implementation of the NMS iteration proposed by Benner and Byers [BB06]. Let us suppose that $\mathcal{H}_k$ in (6.13) is given through a matrix pencil $\mathcal{A}_k - s\mathcal{E}_k$ which is right-similar to it, i.e., $\mathcal{E}_k^{-1}\mathcal{A}_k = \mathcal{H}_k$. Then (6.13) reads

$$\mathcal{H}_{k+1} = \frac{1}{2}\left(\mathcal{E}^{-1}\mathcal{A} + \mathcal{A}^{-1}\mathcal{E}\right).$$

If $(\widetilde{\mathcal{E}_k}, \widetilde{\mathcal{A}_k})$ is a solution of the CS-AB problem for $(\mathcal{E}_k, \mathcal{A}_k)$, i.e., $\widetilde{\mathcal{E}_k}\mathcal{A}_k = \widetilde{\mathcal{A}_k}\mathcal{E}_k$, then direct verification shows that

$$\mathcal{H}_{k+1} = \left(\widetilde{\mathcal{A}_k}\mathcal{E}_k\right)^{-1} \frac{1}{2}\left(\widetilde{\mathcal{A}_k}\mathcal{A}_k + \widetilde{\mathcal{E}_k}\mathcal{E}_k\right),$$

that is, $\mathcal{H}_{k+1}$ is right-similar to

$$s\mathcal{E}_{k+1} - \mathcal{A}_{k+1} := s\widetilde{\mathcal{A}_k}\mathcal{E}_k - \frac{1}{2}\left(\widetilde{\mathcal{A}_k}\mathcal{A}_k + \widetilde{\mathcal{E}_k}\mathcal{E}_k\right). \tag{6.16}$$

The pencil $s\mathcal{E}_{k+1} - \mathcal{A}_{k+1}$ can be computed directly without the need of matrix inversions. The following similarity result is derived with the help of pencil arithmetics in [BB06].

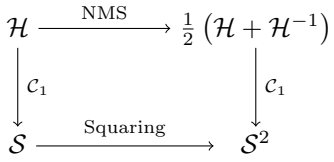


Figure 6.1: Relationship between the NMS iteration, the Cayley transform, and squaring

Theorem 6.15 ([BB06])**.** *Let $\mathcal{H}_0 = \mathcal{A}_0$ be a nonsingular square matrix, and $\mathcal{E}_0 = I$. Then, the sequences $\mathcal{E}_k$, $\mathcal{A}_k$ defined recursively by* (6.16) *are such that $s\mathcal{E}_k - \mathcal{A}_k$ is right-similar to the matrix $\mathcal{H}_k$ obtained by the NMS iteration* (6.13)*.*

By iterating the transformation (6.16) starting from $\mathcal{A}_0 = \mathcal{H}$, $\mathcal{E}_0 = I$, we obtain an implicit version of the NMS iteration. After a sufficient number of iterations, we expect $\mathcal{E}_k^{-1}\mathcal{A}_k$ to converge to $\operatorname{sgn}(\mathcal{H})$. However, the convergence theory is tricky, and not completely settled in [BB06]. Even if there is convergence in a weaker sense regarding deflating subspaces, the two sequences of matrices $\mathcal{E}_k$ and $\mathcal{A}_k$ may diverge, oscillate, converge simultaneously to zero (which is equally problematic), or exhibit different behaviors simultaneously in different subspaces. A suitable normalization of the pencil may be needed in order to ensure convergence of the two sequences, and even in this case it may be ill-conditioned to recover the invariant subspaces from the two limits.

Despite this issues, the inverse-free NMS iteration performs better than its customary counterpart in terms of stability [BB06], especially when the starting matrix $\mathcal{H}$ is ill-conditioned. The price to pay is a higher cost per step.

6.8 Matrix sign and disk iteration

The NMS iteration assumes a simpler form if we make use of a Cayley transform to switch between the continuous-time and discrete-time setting [Ben97]. If we set $\mathcal{S} = \mathcal{C}_1(\mathcal{H})$, then the same Cayley transform verifies

$$\mathcal{S}^2 = \mathcal{C}_1\left(\frac{1}{2}(\mathcal{H}+\mathcal{H}^{-1})\right).$$

In other words, the diagram in Figure 6.1 commutes. This means that the map underlying the NMS iteration and repeated squaring are conjugated by the Cayley transform, and thus iterating one is equivalent to iterating the other [And78, Rob80, Ben97].

This fact is exploited in [Ben97] to replace the iteration (6.13) with the map $\mathcal{S}_{k+1} = \mathcal{S}_k^2$, where $\mathcal{S}_k = \mathcal{C}_1(\mathcal{H}_k)$. The squaring can be performed along the inverse-free framework. Let $\mathcal{S}_k = \mathcal{E}_k^{-1}\mathcal{A}_k$ be given, and $(\widetilde{\mathcal{E}_k}, \widetilde{\mathcal{A}_k})$ be a solution of the CS-AB problem for $(\mathcal{E}_k, \mathcal{A}_k)$. Then,

$$\mathcal{S}_{k+1} = (\widetilde{\mathcal{E}_k}\mathcal{E}_k)^{-1}(\widetilde{\mathcal{A}_k}\mathcal{A}_k). \tag{6.17}$$

Therefore the corresponding inverse-free iteration is

$$s\mathcal{E}_{k+1} - \mathcal{A}_{k+1} = s\widetilde{\mathcal{E}_k}\mathcal{E}_k - \widetilde{\mathcal{A}_k}\mathcal{A}_k, \tag{6.18}$$

which compares favorably in terms of computational cost to (6.16), since it requires one CS-AB solution and only two (instead of three) matrix products. The iteration (6.18) was first used by Malyshev [Mal89] to compute the disk function of a pencil.

A parallel of Theorem 6.15 is easily derived for this iteration.

Theorem 6.16 ([Ben97, BDG97]). *Let $\mathcal{S}_0 = \mathcal{A}_0$ be given and $\mathcal{E}_0 = I$. Then, the sequences $\mathcal{E}_k$, $\mathcal{A}_k$ defined recursively by* (6.18) *are such that $s\mathcal{E}_k - \mathcal{A}_k$ is right-similar to $\mathcal{S}^{2^k}$.*

If this squaring operation is repeated t times, we get to a pencil with the same deflating subspaces of $s\mathcal{E} - \mathcal{A}$, but with eigenvalues given by λ^{2^t}, where λ is an eigenvalue of the original pencil $s\mathcal{E} - \mathcal{A}$. In particular, eigenvalues with $|\lambda| < 1$ converge to 0, and eigenvalues with $|\lambda| > 1$ converge to infinity. Thus we can recover the d-stable and d-unstable deflating subspaces of the initial pencil by computing the deflating subspaces relative to 0 and ∞ of $s\mathcal{E}_t - \mathcal{A}_t$.

This idea is the basis of the inverse-free disk function method [Ben97] reported here in Algorithm 7. Note that, as in the inverse-free NMS method, the numerical stability of the

Algorithm 7: inverse-free disk function method

input: a matrix pencil $s\mathcal{E}_0 - \mathcal{A}_0$
output: U, V bases of the d-stable and d-unstable deflating subspaces of $s\mathcal{E}_0 - \mathcal{A}_0$
  $k \longleftarrow 0$
  while a suitable stopping criterion is not satisfied **do**
    $(\widetilde{\mathcal{E}_k}, \widetilde{\mathcal{A}_k}) \longleftarrow$ any solution of the CS-AB problem for $(\mathcal{A}_k, \mathcal{E}_k)$
    $\mathcal{A}_{k+1} \longleftarrow \widetilde{\mathcal{A}_k}\mathcal{A}_k$
    $\mathcal{E}_{k+1} \longleftarrow \widetilde{\mathcal{E}_k}\mathcal{E}_k$
    $k \longleftarrow k + 1$
  end while
  $U \longleftarrow$ approximate null space of $\mathcal{A}_k$
  $V \longleftarrow$ approximate null space of $\mathcal{E}_k$
  return U, V

algorithm and the convergence of the sequences $\mathcal{A}_k$ and $\mathcal{E}_k$ depend strongly on the choice of the solution to the CS-AB problem.

6.9 The structured doubling algorithm

The structured doubling algorithm (SDA) can be seen as a variant of the inverse-free disk function. It was introduced in an initial form by Anderson [And78] as an algorithm for the solution of a discrete-time algebraic Riccati equation, and later extended, adapted to other equations and explained in terms of matrix pencils in several papers by Wen-Wei Lin and others [CFL05, LX06, HL09, CCG+09].

The main idea is coupling the inverse-free disk iteration with a special form of the initial pencil, which is preserved along the algorithm.

Standard structured form

A pencil $s\mathcal{E} - \mathcal{A}$ with $\mathcal{A}, \mathcal{E} \in \mathbb{R}^{(n+m)\times(n+m)}$ is said to be in *standard structured form I (SSF-I)* if it can be written as

$$\mathcal{A} = \begin{bmatrix} E & 0 \\ -H & I_m \end{bmatrix}, \quad \mathcal{E} = \begin{bmatrix} I_n & -G \\ 0 & F \end{bmatrix}, \tag{6.19}$$

where the block sizes are such that $E \in \mathbb{R}^{n\times n}$ and $F \in \mathbb{R}^{m\times m}$. Note that a pencil in SSF-I is always regular.

Derivation of SDA

With the help of Theorem 6.14, we can build a solution to the CS-AB problem in SSF-I, i.e.,

$$\widetilde{\mathcal{A}} = \begin{bmatrix} \widetilde{E} & 0 \\ -\widetilde{H} & I_m \end{bmatrix}, \quad \widetilde{\mathcal{E}} = \begin{bmatrix} I_n & -\widetilde{G} \\ 0 & \widetilde{F} \end{bmatrix}. \tag{6.20}$$

This is equivalent to imposing

$$\begin{bmatrix} \widetilde{E}_{11} & \widetilde{A}_{12} \\ \widetilde{E}_{21} & \widetilde{A}_{22} \end{bmatrix} = I_{n+m},$$

thus Theorem 6.14 yields

$$\begin{bmatrix} \widetilde{E} & \widetilde{G} \\ \widetilde{H} & \widetilde{F} \end{bmatrix} = \begin{bmatrix} E & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} I_n & -G \\ -H & I_m \end{bmatrix}^{-1},$$

and thus

$$\widetilde{\mathcal{A}} = \begin{bmatrix} E(I_n - GH)^{-1} & 0 \\ -F(I_m - HG)^{-1}H & I_n \end{bmatrix}, \quad \widetilde{\mathcal{E}} = \begin{bmatrix} I_n & -E(I_n - GH)^{-1}G \\ 0 & F(I_m - HG)^{-1} \end{bmatrix}. \tag{6.21}$$

If we apply a step of inverse-free disk iteration (6.18) to $s\mathcal{E} - \mathcal{A}$ using the computed CS-AB solution, the resulting pencil is still in SSF-I, and is given by

$$\begin{aligned} \widehat{\mathcal{A}} &= \begin{bmatrix} E(I_n - GH)^{-1}E & 0 \\ -(H + F(I_m - HG)^{-1}HE) & I_m \end{bmatrix}, \\ \widehat{\mathcal{E}} &= \begin{bmatrix} I_n & -(G + E(I_n - GH)^{-1}GF) \\ 0 & F(I_m - HG)^{-1}F \end{bmatrix}. \end{aligned} \tag{6.22}$$

The only hypothesis required to carry on these operations is that $I - GH$ and $I - HG$ are nonsingular.

Remark 6.17. Let $G \in \mathbb{R}^{n\times m}$, $H \in \mathbb{R}^{m\times n}$. Then $I_n - GH$ is singular if and only if $I_m - HG$ is nonsingular.

Proof. It is a basic fact in linear algebra [Hog07, Chapter 4] that the two products GH and HG have the same spectrum, except for the zero eigenvalues. In particular, GH has the eigenvalue 1 if and only if HG does. □

Algorithm 8 implements this variant of the inverse-free disk iteration (Algorithm 7) by updating directly the blocks of the involved matrices. Each step of the algorithm costs $\frac{14}{3}(m^3 + n^3) + 6mn(m + n)$ floating point operations. This reduces to $\frac{64}{3}n^3$ in the case $m = n$.

Convergence results

The following result summarizes the convergence properties of SDA [CFL05, LX06, HL09, CCG+09].

Theorem 6.18 ([CFL05, LX06, HL09, CCG+09])**.** *Suppose that the pencil* (6.19) *has either a proper or a partial* (n, m) *d-splitting, and that there are two matrices in the form*

$$\begin{bmatrix} I_n \\ H_\infty \end{bmatrix}, \quad \begin{bmatrix} G_\infty \\ I_m \end{bmatrix}, \tag{6.23}$$

Algorithm 8: SDA-I

input: E_0, F_0, G_0, H_0 defining a pencil in SSF-I
output: H_∞,G_∞ so that the subspaces in (6.23) are respectively the canonical semi-d-stable and semi-d-unstable deflating subspaces of the given pencil

$k \longleftarrow 0$
while a suitable stopping criterion is not satisfied **do**
$\widetilde{E_k} \longleftarrow E_k(I_n - G_kH_k)^{-1}$
$\widetilde{F_k} \longleftarrow F_k(I_m - H_kG_k)^{-1}$
$G_{k+1} \longleftarrow G_k + \widetilde{E_k}G_kF_k$
$H_{k+1} \longleftarrow H_k + \widetilde{F_k}H_kE_k$
$E_{k+1} \longleftarrow \widetilde{E_k}E_k$
$F_{k+1} \longleftarrow \widetilde{F_k}F_k$
$k \longleftarrow k+1$
end while
return $H_\infty \longleftarrow H_k$, $G_\infty \longleftarrow G_k$

spanning respectively the canonical d-semi-stable and d-semi-unstable deflating subspace.

Then for Algorithm 8 it holds that

1. $\|E_k\| = O(2^{-k})$,
2. $\|F_k\| = O(2^{-k})$,
3. $\|H_\infty - H_k\| = O(2^{-k})$,
4. $\|G_\infty - G_k\| = O(2^{-k})$.

Moreover, if the splitting is proper, the convergence is quadratic, and in particular in 3. and 4. the term $O(2^{-k})$ can be replaced by $O(\nu^{2^k})$, with ν equal to the dominance factor (2.6) of $\mathcal{H}$.

A new method to compute the SSF-I of a pencil

From the following result, we can derive a compact expression to compute the unique pencil in SSF-I which is right-similar to a given pencil.

Theorem 6.19 ([PR]). *Let $s\mathcal{E} - \mathcal{A}$ be a matrix pencil with $\mathcal{A}, \mathcal{E} \in \mathbb{R}^{(n+m)\times(n+m)}$, and partition both matrices as*

$$\mathcal{A} = \begin{bmatrix}\mathcal{A}_1 & \mathcal{A}_2\end{bmatrix} \quad \mathcal{E} = \begin{bmatrix}\mathcal{E}_1 & \mathcal{E}_2\end{bmatrix}$$

with $\mathcal{A}_1, \mathcal{E}_1 \in \mathbb{R}^{(n+m)\times n}$ and $\mathcal{A}_2, \mathcal{E}_2 \in \mathbb{R}^{(n+m)\times m}$. A SSF-I pencil which is right-similar to $s\mathcal{E} - \mathcal{A}$ exists if and only if

$$\begin{bmatrix}\mathcal{E}_1 & \mathcal{A}_2\end{bmatrix} \tag{6.24}$$

is nonsingular; in this case, it holds

$$\begin{bmatrix} E & -G \\ -H & F \end{bmatrix} = \begin{bmatrix}\mathcal{E}_1 & \mathcal{A}_2\end{bmatrix}^{-1} \begin{bmatrix}\mathcal{A}_1 & \mathcal{E}_2\end{bmatrix}. \tag{6.25}$$

Proof. We are looking for a matrix Q such that

$$Q \begin{bmatrix} \mathcal{A}_1 & \mathcal{A}_2 \end{bmatrix} - sQ \begin{bmatrix} \mathcal{E}_1 & \mathcal{E}_2 \end{bmatrix} = \begin{bmatrix} E & 0 \\ -H & I \end{bmatrix} - s \begin{bmatrix} I & -G \\ 0 & F \end{bmatrix}.$$

By taking only some of the blocks from the above equation, we get

$$Q\mathcal{E}_1 = \begin{bmatrix} I \\ 0 \end{bmatrix}, \quad Q\mathcal{A}_2 = \begin{bmatrix} 0 \\ I \end{bmatrix},$$

i.e.,

$$Q \begin{bmatrix} \mathcal{E}_1 & \mathcal{A}_2 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix},$$

thus Q must be the inverse of the matrix in (6.24).

On the other hand, taking the other two blocks we get

$$Q\mathcal{A}_1 = \begin{bmatrix} E \\ -H \end{bmatrix}, \quad Q\mathcal{E}_2 = \begin{bmatrix} -G \\ F \end{bmatrix},$$

which promptly yields (6.25). □

This formula is connected to the principal pivot transform (PPT) [Tsa00].

SDA for nonsymmetric algebraic Riccati equations

We may solve a NARE (1.5) using SDA with the following steps [GLX06].

- Compute $\mathcal{S} = \mathcal{C}_\gamma(\mathcal{H})$, where $\mathcal{H}$ is as in (1.8), for a suitable $\gamma > 0$. In this way, the c-splitting is transformed into a d-splitting, and the c-unstable subspace is transformed into the d-stable space. Notice that the splitting is partial in the null recurrent case and proper otherwise.
- Find a pencil in SSF-I which is right-similar to $\mathcal{S}$.
- Apply SDA (Algorithm 8) to the resulting pencil.

The two matrices H_∞ and G_∞ returned by SDA are the minimal solution of the NARE (1.5) and of its dual (6.1).

The second step can be performed efficiently thanks to Theorem 6.19. Notice that the Cayley transform of $\mathcal{H}$ is given in pencil form by $(\mathcal{H} - \gamma I) - s(\mathcal{H} + \gamma I)$. Thus (6.25) becomes

$$\begin{bmatrix} E & -G \\ -H & F \end{bmatrix} = \begin{bmatrix} D + \gamma I & -C \\ B & -A - \gamma I \end{bmatrix}^{-1} \begin{bmatrix} D - \gamma I & -C \\ B & -A + \gamma I \end{bmatrix}. \tag{6.26}$$

This formula is more compact to write and more computationally effective than the one suggested by Guo *et al.* [GLX06]. In fact, their expressions for the starting blocks involve computing the inverses of two $n \times n$ and $m \times m$ matrices, which are indeed the $(1,1)$ and $(2,2)$ blocks of the matrix to be inverted in (6.26) and their Schur complements. Clearly, two these inversions are redundant in (6.26), which requires more or less half of the computational cost with respect to the original formulas in [GLX06].

Moreover, the parameter γ can be chosen in order to guarantee applicability and convergence of SDA.

Theorem 6.20 ([GIM07]). *Let $\mathcal{M}$ be irreducible and γ be chosen such that*

$$\gamma \geqslant \max\left\{\max_{1\leq i\leq m} A_{ii}, \max_{1\leq j\leq n} D_{jj}\right\}. \tag{6.27}$$

Then for each $k \geq 0$ the iterates E_k, F_k, G_k H_k generated by SDA are nonnegative matrices (except for $E_0, F_0 \leq 0$, and the matrices $I - G_k H_k$ and $I - H_k G_k$ are nonsingular M-matrices. Moreover, the sequences G_k and H_k converge monotonically.

6.10 Conclusions and research lines

This chapter presents an overview of the theoretical properties and numerical algorithms for the solution of the NARE (1.5), which we also presented in the review work [BIMP10]. The main difference between the two is the treatment of SDA, which is introduced here with a more natural derivation as a modification of the inverse-free disk and NMS functions. Theorem 6.13 provides a new way to generate solutions of the CS-AB problem, and Theorem 6.19 a simpler method to compute the SSF-I of a pencil than those existing in literature.

An important research problem is incorporating one of the scaling strategies used in NMS iteration [Hig08] into SDA. As is shown by Higham [Hig08], scaling can have a large impact on the convergence speed of the NMS iteration (and thus of SDA). Unfortunately, it is not straightforward how to implement such a scaling, since the scaling in the NMS setting becomes a more involved function in the squaring/SDA setting after a Cayley transform. We have obtained several encouraging partial results using Theorem 6.19 to recompute the SSF-I factorization after a scaling transformation, but the issue has not been settled completely.

There are other problems for which SDA could possibly be applied successfully, and that we would like to investigate further. We list two of them.

- Matrix sign function computation (instead of the NMS iteration [Hig08, BB06]). Apart from the scaling, the other issue is that SDA apparently needs to know in advance how many stable and unstable eigenvalues the starting matrix has. On the other hand, the application of Theorem 6.19 could be the key to providing a method to change the block sizes of SDA dynamically, e.g., from (m, n) to $(m + k, n - k)$.

- Multiplication-rich divide-and-conquer spectral division algorithm for high-performance computing. Currently, several papers [Mal89, BDG97, DDH07] advocate using the inverse-free NMS and disk functions to this purpose. SDA would make a good tool for this problem: in fact, it should be easier with SDA than with inverse-free NMS to enforce an even splitting of the spectrum, which boosts the performance of divide-and-conquer.

For the control theory applications, a criterion for the choice of the parameter γ for the Cayley transform similar to that of Theorem 6.20 does not exist, up to our knowledge. In Section 10.4 we address this problem and try to find a heuristic for its choice.

The shift technique [HMR02, GIM07] is a tool to speed up the convergence in presence of a singular $\mathcal{M}$ by modifying it in order to remove the zero eigenvalue and increase the dominance factor. We are studying [IP] a modification of the shift technique to deal in the same way with close-to-singular problems.

Chapter 7

Transforming NAREs into UQMEs

7.1 Introduction

The problem of reducing an algebraic Riccati equation (1.5) to a unilateral quadratic matrix equation (1.2) has been considered in [BILM06, GLX06, IB03, LR80, Ram99].

In this chapter we introduce two methods to transform the ARE into a UQME [BMP10b]. These transformations enable us to prove some theoretical and computational properties. In particular they provide a unifying framework which includes apparently different algorithms like the algorithm of Ramaswami [Ram99] and the structure preserving doubling algorithm (SDA) of Anderson [And78], and Guo, Lin, Xu [GLX06]. We show that SDA can be interpreted as cyclic reduction algorithm applied to a suitable UQME. This fact enables one to deduce some of the convergence properties of SDA directly from the theory of cyclic reduction which is well consolidated [BLM05, BMR08, BM09]. The result has an interesting counterpart in [CCG$^+$09], where, on the other hand, it is proved that cyclic reduction can be interpreted as a form of SDA with a block structure which is different from SSF-I (6.19).

This unifying framework allows us to design some new algorithms for the effective solution of an ARE. In particular, by combining the SDA idea with the shrink-and-shift technique of Ramaswami we obtain a new algorithm having the same cost per iteration as SDA but relying on a different and simpler initialization. We prove that this algorithm has better convergence properties with respect to SDA when $\mathcal{M}$ is an M-matrix and $\max a_{i,i}/\max d_{i,i}$ is very large or very small as in the fluid queues models of [BOT05]. In fact, for the equations of [BOT05] the speedup in terms of iterations is by a factor greater than 4, as shown in our numerical experiments.

The transformations are based on two steps, namely, transforming the matrix pencil $sI - \mathcal{H}$ to a structured quadratic matrix polynomial and applying a suitable matrix function to convert the c-splitting of the Hamiltonian (1.8) into a d-splitting.

After these steps, we arrive at a UQME (1.2) whose solutions are simply related to the solutions of (1.5). Due to the d-splitting, cyclic reduction can be applied for the solution of this UQME. The block structure of the coefficients of the UQME allows to implement cyclic reduction with a lower computational cost.

7.2 Assumptions on Algebraic Riccati Equations

The transformations we present work under general assumptions on the NARE. This makes them suitable not only for the nonsymmetric equations associated with an M-matrix, but also for their symmetric counterpart in control theory and for more general complex problems.

In cases where more structure is present (Hamiltonian and symmetric structure, or nonnegativity and M-matrix structure), this can be used to prove that CR converges without breakdown and preserving said structure, but the general hypotheses needed for the transformation are not very restrictive.

Throughout this chapter, unless differently specified, we assume the following.

Assumption 7.1 (c-splitting)**.** The eigenvalues of $\mathcal{H}$ are such that

$$\operatorname{Re}\lambda_{n+1} \leq 0 \leq \operatorname{Re}\lambda_n. \tag{7.1}$$

Moreover, the invariant subspace of $\mathcal{H}$ corresponding to the eigenvalues $\lambda_1, \ldots, \lambda_n$ is spanned by a matrix in the form (6.2), with U_1 nonsingular. This is equivalent to assuming the existence of a solution $S = U_2 U_1^{-1}$ such that $\sigma(D - CS) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. Similarly, the invariant subspace corresponding to $\lambda_{n+1}, \lambda_{n+2}, \ldots, \lambda_{n+m}$ is spanned by a matrix in the form

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}, \qquad \text{with } V_1 \in \mathbb{R}^{n\times m},\ V_2 \in \mathbb{R}^{m\times m},$$

with V_2 nonsingular. This is equivalent to assuming the existence of a solution $T = V_1 V_2^{-1}$ to the dual equation (6.1) such that $\sigma(A - BT) = \{\lambda_{n+1}, \lambda_{n+2}, \ldots, \lambda_{n+m}\}$.

Assumption 7.1 is satisfied in particular under the following stronger conditions which are encountered in diverse applications.

Assumption 7.1.1 (Symmetric case [LR95, Meh91])**.** The matrix $\mathcal{H}$ is such that $D = A^T$, C and $-B$ are symmetric, positive-definite, the pair (D, C) is stabilizable and the pair (B, D) is detectable.

Assumption 7.1.2 (M-matrix case [BOT05, Guo01, Ram99])**.** The matrix $\mathcal{M}$ of (1.6) is either a nonsingular M-matrix or a singular irreducible M-matrix.

Assumption 7.1.3 (Complex case [Guo07])**.** The matrix $\mathcal{H}$ is complex and such that $\widetilde{\mathcal{H}} = (\widetilde{h}_{i,j})$, $\widetilde{h}_{i,i} = |h_{i,i}|$, $\widetilde{h}_{i,j} = -|h_{i,j}|$, if $i \neq j$, is an M-matrix, moreover the diagonal elements of $\mathcal{H}$ are real and either all positive or all negative.

In the following we restrict our analysis to the case where the condition $\Re(\lambda_n) = \Re(\lambda_{n+1}) = 0$ is not satisfied. We recall that if $\lambda_n = \lambda_{n+1} = 0$, then one can apply suitable techniques in order to overcome the difficulties encountered in this critical case [GIM07, BIMP10].

Thanks to Lemma 6.4, in the sequel we may assume the following without loss of generality.

Assumption 7.2. The eigenvalues of $\mathcal{H}$, ordered as in (6.5), are such that $\Re(\lambda_{n+1}) < 0 \leq \Re(\lambda_n)$.

7.3 Transformations of a NARE into a UQME

In this section we present two methods to convert a NARE (1.5) into an equivalent UQME (1.2). These transformations are based on the idea of modifying the pencil $sI - \mathcal{H}$ to get a quadratic matrix polynomial having the same eigenvalues plus some additional ones at zero and at infinity.

Ramaswami's transformation

Ramaswami [Ram99] proposed a method to transform the NARE into a UQME of the kind

$$\mathcal{A}_0 + \mathcal{A}_1\mathcal{X} + \mathcal{A}_2\mathcal{X}^2 = 0, \quad \mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{X} \in \mathbb{R}^{(m+n)\times(m+n)}.$$

His idea can be interpreted in the following way. The eigenvalue problem

$$0 = (\mathcal{H} - sI)\, u = \left(\begin{bmatrix} D & -C \\ B & -A \end{bmatrix} - s \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \right) u$$

originating from (1.7) is transformed into a quadratic eigenvalue problem by multiplying the second block column by s:

$$\left(\begin{bmatrix} D & 0 \\ B & 0 \end{bmatrix} + \begin{bmatrix} -I & -C \\ 0 & -A \end{bmatrix} s + \begin{bmatrix} 0 & 0 \\ 0 & -I \end{bmatrix} s^2 \right) \widehat{u} = 0, \quad \widehat{u} = \begin{bmatrix} I & 0 \\ 0 & s^{-1}I \end{bmatrix} u.$$

With the latter quadratic pencil we may associate the UQME

$$\begin{bmatrix} D & 0 \\ B & 0 \end{bmatrix} + \begin{bmatrix} -I & -C \\ 0 & -A \end{bmatrix} \mathcal{X} + \begin{bmatrix} 0 & 0 \\ 0 & -I \end{bmatrix} \mathcal{X}^2 = 0, \tag{7.2}$$

defined by the matrix quadratic polynomial

$$\begin{gathered} \mathcal{A}(s) = \mathcal{A}_0 + s\mathcal{A}_1 + s^2\mathcal{A}_2, \\ \mathcal{A}_0 = \begin{bmatrix} D & 0 \\ B & 0 \end{bmatrix}, \quad \mathcal{A}_1 = \begin{bmatrix} -I & -C \\ 0 & -A \end{bmatrix}, \quad \mathcal{A}_2 = \begin{bmatrix} 0 & 0 \\ 0 & -I \end{bmatrix}. \end{gathered} \tag{7.3}$$

The eigenvalues of $\mathcal{A}(s)$ and the eigenvalues of $\mathcal{H}$, as well as the solutions of (1.5) and the UQME (7.2), are closely related as stated by the following theorem.

Theorem 7.1 ([Ram99, BMP10b])**.** *The eigenvalues of the matrix polynomial $\mathcal{A}(s)$ defined in* (7.3) *are:*

- 0 *with multiplicity m,*
- *the $m+n$ (counted with multiplicity) eigenvalues $\lambda_1, \ldots, \lambda_{m+n}$ of $\mathcal{H}$,*
- ∞ *with multiplicity n.*

Moreover, if X is a solution of the NARE (1.5)*, then*

$$\mathcal{X} = \begin{bmatrix} D - CX & 0 \\ X & 0 \end{bmatrix}$$

is a solution to the UQME (7.2)*; conversely,*

$$\mathcal{S} = \begin{bmatrix} D - CS & 0 \\ S & 0 \end{bmatrix} \tag{7.4}$$

where S is the extremal solution of (1.5)*, is the unique solution of the UQME* (7.2) *with m eigenvalues equal to zero and n eigenvalues equal to $\lambda_1, \ldots, \lambda_n$.*

Proof. By construction, one has

$$\mathcal{A}(s) = (\mathcal{H} - sI_{m+n}) \begin{bmatrix} I_n & 0 \\ 0 & sI_m \end{bmatrix}.$$

Therefore $\det \mathcal{A}(s) = s^m \det(\mathcal{H} - sI_{m+n})$ and the properties of the eigenvalues follow. If X solves the NARE (1.5), one may verify by direct inspection that $\mathcal{X}$ solves the UQME (7.2). In particular $\mathcal{S}$ is a solution of (7.2), and its eigenvalues are zero with multiplicity m and $\lambda_1, \ldots, \lambda_n$; for the properties of the roots of $\mathcal{A}(s)$, $\mathcal{S}$ is the unique solution with these eigenvalues. □

UL-based transformation

We introduce another transformation of a NARE to a UQME which relies on the block UL factorization of the matrix $\mathcal{H}$. As we discuss in Section 7.5, this transformation is implicitly used in the SDA, and allows to relate SDA to CR.

Let us consider the block UL factorization

$$\mathcal{H} = \mathcal{U}^{-1}\mathcal{L}, \qquad \mathcal{U} = \begin{bmatrix} I & -U_1 \\ 0 & U_2 \end{bmatrix}, \qquad \mathcal{L} = \begin{bmatrix} L_1 & 0 \\ -L_2 & I \end{bmatrix},$$

with

$$U_1 = CA^{-1}, \qquad U_2 = -A^{-1}, \qquad L_1 = D - CA^{-1}B, \qquad L_2 = A^{-1}B,$$

where we assume $\det A \neq 0$, and transform the eigenvalue problem

$$0 = (\mathcal{H} - sI)u = (\mathcal{U}^{-1}\mathcal{L} - sI)u$$

into the generalized one $(\mathcal{L} - s\mathcal{U})u = 0$ via a right-similarity. Now we multiply the second block row by $-s$ to get

$$\left(\begin{bmatrix} L_1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -I & U_1 \\ L_2 & -I \end{bmatrix} s + \begin{bmatrix} 0 & 0 \\ 0 & U_2 \end{bmatrix} s^2 \right) u = 0.$$

As in the previous section, with the latter quadratic pencil we may associate the UQME

$$\begin{bmatrix} L_1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -I & U_1 \\ L_2 & -I \end{bmatrix} \mathcal{X} + \begin{bmatrix} 0 & 0 \\ 0 & U_2 \end{bmatrix} \mathcal{X}^2 = 0, \tag{7.5}$$

defined by the matrix quadratic polynomial

$$\begin{gathered} \mathcal{A}(s) = \mathcal{A}_0 + s\mathcal{A}_1 + s^2\mathcal{A}_2, \\ \mathcal{A}_0 = \begin{bmatrix} L_1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathcal{A}_1 = \begin{bmatrix} -I & U_1 \\ L_2 & -I \end{bmatrix}, \quad \mathcal{A}_2 = \begin{bmatrix} 0 & 0 \\ 0 & U_2 \end{bmatrix}. \end{gathered} \tag{7.6}$$

Then the following result holds.

Theorem 7.2 ([BMP10b])**.** *The eigenvalues of the matrix polynomial $\mathcal{A}(s)$ defined in* (7.6) *are:*

- 0 *with multiplicity m,*
- *the $m + n$ (counted with multiplicity) eigenvalues $\lambda_1, \ldots, \lambda_{m+n}$ of $\mathcal{H}$,*

- ∞ *with multiplicity* n.

Moreover, if X *is a solution of the NARE* (1.5)*, then*

$$\mathcal{X} = \begin{bmatrix} D - CX & 0 \\ X(D - CX) & 0 \end{bmatrix}$$

is a solution to the UQME (7.5)*. Conversely,*

$$\mathcal{S} = \begin{bmatrix} D - CS & 0 \\ S(D - CS) & 0 \end{bmatrix},$$

where S *is the extremal solution of* (1.5)*, is the unique solution of the UQME* (7.2) *with* m *eigenvalues equal to zero and* n *eigenvalues equal to* $\lambda_1, \ldots, \lambda_n$.

Proof. By construction, one has

$$\mathcal{A}(s) = \begin{bmatrix} I_n & 0 \\ 0 & -sI_m \end{bmatrix} \mathcal{U}(\mathcal{H} - sI_{m+n}).$$

Therefore $\det \mathcal{A}(s) = (-1)^m s^m \det \mathcal{U} \det(\mathcal{H} - sI_{m+n})$ and the spectral properties follow. The remaining part can be proved as in Theorem 7.1. □

7.4 Eigenvalue transformations

The transformations presented in Section 7.3 can effectively be used to transform the NARE into a UQME. The solution of interest in NAREs is the extremal one, which is the one associated with the c-semi-unstable eigenvalues. Algorithms for solving UQMEs are usually designed to find the solution with d-semi-stable eigenvalues; therefore, we need to apply an eigenvalue transformation in order to transform the c-splitting to a d-splitting. Several strategies are available for this task; the basic idea is the following lemma.

Lemma 7.3. *Let* $a(s) = a_0 + a_1 s + \ldots + a_h s^h$ *and* $b(s) = b_0 + b_1 s + \ldots + b_l s^l$ *be polynomials with complex coefficients, and extend them to square matrices as* $a(X) = a_0 I + a_1 X + \ldots + a_h X^h$ *and* $b(X) = b_0 I + b_1 X + \ldots + b_l X^l$*. Let* $f(s) = \frac{a(s)}{b(s)}$ *and* $f(X) = b(X)^{-1} a(X)$*. Let* λ_i *be the eigenvalues of the matrix* $\mathcal{H}$*. If* $b(\lambda_i) \neq 0$ *for each* i*, then the eigenvalues of* $f(\mathcal{H})$ *are* $f(\lambda_i)$*. Moreover, if*

$$\mathcal{H} \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} R \tag{7.7}$$

holds, then

$$f(\mathcal{H}) \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} f(R) \tag{7.8}$$

holds as well.

Proof. The first part is well-known [HJ94]. As for the last formula, by applying repeatedly (7.7) we get

$$\mathcal{H}^k \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} R^k \quad \text{for all } k \geqslant 0; \tag{7.9}$$

then we rewrite (7.8) as

$$a(\mathcal{H}) \begin{bmatrix} I \\ X \end{bmatrix} b(R) = b(\mathcal{H}) \begin{bmatrix} I \\ X \end{bmatrix} a(R)$$

(observe that $a(R)$ and $b(R)$ commute), and apply (7.9) to all the monomials appearing in the above expression. □

In particular, this result implies that the solutions of the NARE associated with $\mathcal{H}$ are the same as the ones of the NARE associated with $f(\mathcal{H})$, for each rational function f for which $b(\mathcal{H})$ is nonsingular.

Shrink and shift

Ramaswami's approach [Ram99] for the eigenvalue transformation is based on a shrink-and-shift approach. Under Assumption 7.2, if there is no eigenvalue on the imaginary axis, except for zero, then for a sufficiently large value of $t > 0$ one has $|t - \lambda_i| \leq t$ for $i = 1, \ldots, n$, that is, the eigenvalues of $\mathcal{H}$ with nonnegative real part are contained in a sufficiently large disk $\mathcal{D}$ of center t and radius t. The remaining eigenvalues clearly lie outside $\mathcal{D}$, since they are on the opposite side of the imaginary axis. The transformation $f_t : z \mapsto 1 - \frac{z}{t}$ maps $\mathcal{D}$ onto the unit disk $|z| \leq 1$. Applying to $\mathcal{H}$ the transformation

$$\mathcal{H} \mapsto f_t(\mathcal{H}) = I - \frac{1}{t}\mathcal{H},$$

yields the matrix

$$\widehat{\mathcal{H}} := f(\mathcal{H}) = \begin{bmatrix} \widehat{D} & -\widehat{C} \\ \widehat{B} & -\widehat{A} \end{bmatrix},$$

where

$$\widehat{A} := -I - \frac{1}{t}A, \qquad \widehat{B} := -\frac{1}{t}B, \qquad \widehat{C} := -\frac{1}{t}C, \qquad \widehat{D} := I - \frac{1}{t}D. \tag{7.10}$$

By Lemma 7.3, the eigenvalues of $f_t(\mathcal{H})$ are d-split. More precisely, under Assumption 7.2 for the eigenvalues $\mu_i = f_t(\lambda_i)$ of $f_t(\mathcal{H})$ we have

$$|\mu_i| \leq 1 \quad \text{for } i = 1, \ldots, n, \quad |\mu_i| > 1 \quad \text{for } i = n+1, \ldots, n+m,$$

By the same lemma, the solutions to the NARE associated with $f(\mathcal{H})$ are the same as the solutions of the original NARE (1.5).

As to the choice of t, with Assumption 7.1.2 it is sufficient to take

$$t \geqslant \max_{1 \leq i \leq n} D_{ii}. \tag{7.11}$$

In fact, with this choice of t the M-matrix $D - CS$ can be put in the form $tI - P$, with P a nonnegative matrix; thus $\rho(P) \leq t$, with equality only when $D - CS$ is singular. Therefore all the eigenvalues λ of $D - CS$ satisfy $|\lambda - t| \leq t$.

Cayley transform

Another approach [LR95, GLX06, BILM06] is applying the Cayley transform (2.5) with $\gamma > 0$. By Lemma 2.10, transforming

$$\mathcal{H} \mapsto \mathcal{H}_\gamma := \mathcal{C}_\gamma(\mathcal{H}) = (\mathcal{H} + \gamma I)^{-1}(\mathcal{H} - \gamma I)$$

maps the n eigenvalues of $\mathcal{H}$ lying in the right half-plane into the closed unit disk, and the other m eigenvalues to the outside of it. More precisely, under Assumption 7.2, the eigenvalues of $\mathcal{H}_\gamma$ are $\xi_i = \mathcal{C}_\gamma(\lambda_i)$, $i = 1, \ldots, m+n$, and are such that

$$\max_{i=1,\ldots,n} |\xi_i| \leq 1 < \min_{i=1,\ldots,m} |\xi_{i+n}|.$$

Moreover, the solutions of the Riccati equation (1.5) are solutions of the Riccati equation associated with $\mathcal{H}_\gamma$, according to Lemma 7.3.

Observe that the blocks of $\mathcal{H}_\gamma$ are

$$\mathcal{H}_\gamma = \mathcal{C}_\gamma(\mathcal{H}) = \begin{bmatrix} \widetilde{D} & -\widetilde{C} \\ \widetilde{B} & -\widetilde{A} \end{bmatrix},$$

where

$$\begin{aligned} \widetilde{A} &:= -I + 2\gamma V^{-1}, & \widetilde{B} &:= 2\gamma(-A+\gamma I)^{-1}BW^{-1}, \\ \widetilde{C} &:= 2\gamma(D+\gamma I)^{-1}CV^{-1}, & \widetilde{D} &:= I - 2\gamma W^{-1}, \end{aligned} \tag{7.12}$$

with $V := -A + \gamma I + B(D+\gamma I)^{-1}C$ and $W := D + \gamma I + C(-A+\gamma I)^{-1}B$.

7.5 Old and new algorithms

The algorithms that we outline in this section are based on two main steps.

In the first step, the Hamiltonian $\mathcal{H}$ is transformed into a matrix $\widetilde{\mathcal{H}}$ whose eigenvalues are split with respect to the unit circle. This can be obtained either by means of the shrink-and-shift technique of Section 7.4, or by means of the Cayley transform of Section 7.4. According to Lemma 7.3 the solutions of the Riccati equations associated with $\mathcal{H}$ and $\widetilde{\mathcal{H}}$ are the same; in particular, the extremal solution S of the NARE (1.5) is the solution of the NARE associated with $\widetilde{\mathcal{H}}$ corresponding to the eigenvalues in the unit disk.

In the second step, the NARE associated with $\widetilde{\mathcal{H}}$ is transformed into a UQME. This is achieved by means of one of the two techniques of Section 7.3. The resulting UQME is solved by means of cyclic reduction.

According to the combination of the techniques used for performing the above two steps we obtain known and new algorithms.

Shrink-and-shift with Ramaswami's transformation

Under Assumption 7.1.2, Ramaswami [Ram99] and later Guo [Guo06] proposed two similar algorithms based on logarithmic reduction [LR93]. First, one performs the shrink-and-shift transformation defined in Section 7.4, to get $\widehat{\mathcal{H}} = I - \frac{1}{t}\mathcal{H}$. Then, one applies Ramaswami's transformation defined in Section 7.3 to the NARE associated with the Hamiltonian $\widehat{\mathcal{H}}$, to get an $(n+m)\times(n+m)$ UQME whose coefficients have the following block sparsity pattern

$$\begin{bmatrix} * & 0 \\ * & 0 \end{bmatrix} + \begin{bmatrix} -I & * \\ 0 & * \end{bmatrix} \mathcal{X} + \begin{bmatrix} 0 & 0 \\ 0 & * \end{bmatrix} \mathcal{X}^2 = 0. \tag{7.13}$$

Ramaswami [Ram99] and Guo [Guo06] applied logarithmic reduction to the above UQME. Here we apply cyclic reduction, which has a slightly lower computational cost per step as compared to logarithmic reduction [BLM05]. It is easy to see that the sparsity pattern of the block coefficients in (7.13) is preserved during the iterations of cyclic reduction; therefore, CR can be accelerated by working only on "small" blocks and removing the products involving zero blocks.

More precisely, it turns out that applying cyclic reduction to the equation (7.2), where $\mathcal{H}$ is replaced by $\widehat{\mathcal{H}}$, yields blocks of the kind

$$\mathcal{A}_0^{(k)} = \begin{bmatrix} R_1^{(k)} & 0 \\ R_2^{(k)} & 0 \end{bmatrix}, \qquad \mathcal{A}_1^{(k)} = \begin{bmatrix} -I & R_3^{(k)} \\ R_4^{(k)} & R_5^{(k)} \end{bmatrix},$$
$$\mathcal{A}_2^{(k)} = \begin{bmatrix} 0 & 0 \\ 0 & R_6^{(k)} \end{bmatrix}, \qquad \widehat{A}^{(k)} = \begin{bmatrix} -I & R_3^{(0)} \\ R_4^{(k)} & R_5^{(0)} \end{bmatrix}.$$

It can be verified that the matrices $R_i^{(k)}$, $i = 1, \ldots, 6$, satisfy the following equations:

$$\begin{aligned} S^{(k)} &= R_5^{(k)} + R_4^{(k)} R_3^{(k)}, \\ Y^{(k)} &= \left(S^{(k)}\right)^{-1} \left(R_2^{(k)} + R_4^{(k)} R_1^{(k)}\right), \\ X^{(k)} &= R_3^{(k)} Y^{(k)} - R_1^{(k)}, \\ Z^{(k)} &= \left(S^{(k)}\right)^{-1} R_6^{(k)}, \\ T^{(k)} &= R_3^{(k)} Z^{(k)}, \end{aligned} \qquad \begin{aligned} R_1^{(k+1)} &= -R_1^{(k)} X^{(k)}, \\ R_2^{(k+1)} &= -R_2^{(k)} X^{(k)}, \\ R_3^{(k+1)} &= R_3^{(k)} - R_1^{(k)} T^{(k)}, \\ R_4^{(k+1)} &= R_4^{(k)} - R_6^{(k)} Y^{(k)}, \\ R_5^{(k+1)} &= R_5^{(k)} - R_2^{(k)} T^{(k)}, \\ R_6^{(k+1)} &= -R_6^{(k)} Z^{(k)}. \end{aligned}$$

for $k = 0, 1, \ldots,$ starting from the initial values $R_1^{(0)} = I - \frac{1}{t} D$, $R_2^{(0)} = -\frac{1}{t} B$, $R_3^{(0)} = \frac{1}{t} C$, $R_4^{(0)} = 0$, $R_5^{(0)} = I + \frac{1}{t} A$, $R_6^{(0)} = -I$. This way, the CR iteration requires 12 matrix products and one LU factorization per step, leading to a total cost of $\frac{74}{3} n^3$ ops per step (when $m = n$).

From Theorem 5.3 and from Theorem 7.1 applied to $\mathcal{H} = \widetilde{\mathcal{H}}$ it follows that

$$S = -\left(R_5^{(0)} + R_4^{(k)} R_3^{(0)}\right)^{-1} \left(R_2^{(0)} + R_4^{(k)} R_1^{(0)}\right) + O(\nu^{2^k}),$$

where

$$\nu := \frac{\max_{i=1,\ldots,n} |\mu_i|}{\min_{i=1,\ldots,m} |\mu_{n+i}|} < 1,$$

and $\mu_i = f_t(\lambda_i) = 1 - \frac{1}{t}\lambda_i$ for $i = 1, \ldots, m + n$.

Cayley transform with UL-based transformation

A different approach consists in applying the Cayley transform followed by the UL-based transformation of Section 7.3. The following result shows that SDA is in fact CR applied to the resulting UQME.

Theorem 7.4 ([BMP10b])**.** *Let*

$$\begin{bmatrix} L_1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -I & U_1 \\ L_2 & -I \end{bmatrix} \mathcal{X} + \begin{bmatrix} 0 & 0 \\ 0 & U_2 \end{bmatrix} \mathcal{X}^2 = 0 \tag{7.14}$$

be the UQME obtained by applying the Cayley transform to $\mathcal{H}$ followed by the UL-based transformation of an ARE to a UQME. Then

$$L_1 = E, \quad L_2 = H, \quad U_1 = G, \quad U_2 = F,$$

where the matrices E, F, G, H are defined by (6.26)*. Moreover the matrix sequences $\mathcal{A}_i^{(k)}$, $i = 0, 1, 2$, generated by CR (Algorithm 4) applied to* (7.14) *are given by*

$$\mathcal{A}_0^{(k)} = \begin{bmatrix} E_k & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathcal{A}_1^{(k)} = \begin{bmatrix} -I & G_k \\ H_k & -I \end{bmatrix}, \quad \mathcal{A}_2^{(k)} = \begin{bmatrix} 0 & 0 \\ 0 & F_k \end{bmatrix},$$

where $\{E_k\}$, $\{F_k\}$, $\{G_k\}$ and $\{H_k\}$ are the sequences generated by SDA (Algorithm 8).

Proof. The application of the Cayley transform to $\mathcal{H}$ generates a matrix $\mathcal{H}_\gamma = \mathcal{C}_\gamma(\mathcal{H})$ which can be factored as

$$\mathcal{H}_\gamma = \mathcal{U}^{-1}\mathcal{L} = \begin{bmatrix} I & -G \\ 0 & F \end{bmatrix}^{-1} \begin{bmatrix} E & 0 \\ -H & I \end{bmatrix}, \tag{7.15}$$

where E, F, G, H are the initial values of SDA. Applying the transformation of Section 7.3 leads to the UQME (7.14). We can see that applying CR to this equation preserves the sparsity pattern of the coefficients $\mathcal{A}_i$, that is, the iterates can be written as

$$\mathcal{A}_0^{(k)} = \begin{bmatrix} E_k & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathcal{A}_1^{(k)} = \begin{bmatrix} -I & G_k \\ H_k & -I \end{bmatrix}, \quad \mathcal{A}_2^{(k)} = \begin{bmatrix} 0 & 0 \\ 0 & F_k \end{bmatrix}$$

for suitable matrices E_k, F_k, G_k, H_k. Carrying out the CR iteration (Algorithm 4) block by block leads to exactly the same relations (6.22) that define SDA. □

In the following theorem, we provide an explicit expression for the solutions of several unilateral equations related to this transformation.

Theorem 7.5 ([BMP10b])**.** *Let*

$$\mathcal{A}_0 = \begin{bmatrix} E & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathcal{A}_1 = \begin{bmatrix} -I & G \\ H & -I \end{bmatrix}, \quad \mathcal{A}_2 = \begin{bmatrix} 0 & 0 \\ 0 & F \end{bmatrix},$$

where the matrices E, F, G, H are given by (6.26)*, and let*

$$\varphi(s) = s^{-1}\mathcal{A}_0 + \mathcal{A}_1 + sA_1.$$

Define

$$\mathcal{S} = \begin{bmatrix} R_\gamma & 0 \\ SR_\gamma & 0 \end{bmatrix}, \qquad \mathcal{T} = \begin{bmatrix} 0 & TU_\gamma \\ 0 & U_\gamma \end{bmatrix}, \qquad \widehat{\mathcal{S}} = \begin{bmatrix} 0 & 0 \\ Q_\gamma S & Q_\gamma \end{bmatrix}, \qquad \widehat{\mathcal{T}} = \begin{bmatrix} P_\gamma & P_\gamma T \\ 0 & 0 \end{bmatrix},$$

where $R_\gamma = \mathcal{C}_\gamma(R)$ is the Cayley transform of $R = D - CS$, $U_\gamma = \mathcal{C}_\gamma(U)$ is the Cayley transform of $U = BT - A$, $Q_\gamma = F(I - SG)^{-1}$, $P_\gamma = E(I - TH)^{-1}$. Then:

1. *the matrix $\mathcal{S}$ is the only solution to*

$$\mathcal{A}_0 + \mathcal{A}_1\mathcal{X} + \mathcal{A}_2\mathcal{X}^2 = 0 \tag{7.16}$$

   such that $\rho(\mathcal{S}) \leq 1$;

2. *the matrix $\widehat{\mathcal{S}}$ is the only solution to*

$$\mathcal{A}_2 + \mathcal{X}\mathcal{A}_1 + \mathcal{X}^2\mathcal{A}_0 = 0 \tag{7.17}$$

   such that $\rho(\widehat{\mathcal{S}}) < 1$;

3. *the matrix* $\mathcal{T}$ *is the only solution to*

$$\mathcal{A}_2 + \mathcal{A}_1\mathcal{X} + \mathcal{A}_0\mathcal{X}^2 = 0 \tag{7.18}$$

such that $\rho(\mathcal{T}) \leq 1$*;*

4. *the matrix* $\widehat{\mathcal{T}}$ *is the only solution to*

$$\mathcal{A}_0 + \mathcal{X}\mathcal{A}_1 + \mathcal{X}^2\mathcal{A}_2 = 0 \tag{7.19}$$

such that $\rho(\widehat{\mathcal{T}}) < 1$*;*

5. *the following* canonical factorizations *hold for* $|s| = 1$

$$\varphi(s) = (I - s\widehat{\mathcal{S}})\mathcal{W}(I - s^{-1}\mathcal{S}), \quad \varphi(s) = (I - s^{-1}\widehat{\mathcal{T}})\mathcal{Z}(I - s\mathcal{T}), \tag{7.20}$$

where

$$\mathcal{W} = \mathcal{A}_2\mathcal{S} + \mathcal{A}_1 = \begin{bmatrix} -I & G \\ S & -I \end{bmatrix}, \quad \mathcal{Z} = \mathcal{A}_0\mathcal{T} + \mathcal{A}_1 = \begin{bmatrix} -I & T \\ H & -I \end{bmatrix}.$$

Proof. The matrix $\mathcal{H}_\gamma = \mathcal{C}_\gamma(\mathcal{H})$ has eigenvalues $\xi_i = \mathcal{C}_\gamma(\lambda_i)$, which are split with respect to the unit circle. From Lemma 7.3 one has $\mathcal{H}_\gamma \begin{bmatrix} I \\ S \end{bmatrix} = \begin{bmatrix} I \\ S \end{bmatrix} R_\gamma$, which in view of (7.15) is equivalent to

$$\begin{bmatrix} E & 0 \\ -H & I \end{bmatrix} \begin{bmatrix} I \\ S \end{bmatrix} = \begin{bmatrix} I & -G \\ 0 & F \end{bmatrix} \begin{bmatrix} I \\ S \end{bmatrix} R_\gamma. \tag{7.21}$$

From Theorem 7.2 applied to $\mathcal{H}_\gamma$ it follows that the matrix $\mathcal{S}$ is the only solution to (7.16) with eigenvalues $\xi_1, \ldots, \xi_n$, so that $\rho(\mathcal{S}) \leq 1$. It can be easily verified by direct inspection that the matrix $\widehat{\mathcal{S}} = -\mathcal{A}_2\mathcal{W}^{-1}$, for $\mathcal{W} = \mathcal{A}_1 + \mathcal{A}_2\mathcal{S}$, solves (7.17). By using the structure of $\mathcal{A}_1$ and $\mathcal{A}_2$ we find that

$$\mathcal{W} = \begin{bmatrix} 0 & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} R_\gamma & 0 \\ SR_\gamma & 0 \end{bmatrix} + \begin{bmatrix} -I & G \\ H & -I \end{bmatrix} = \begin{bmatrix} -I & G \\ S & -I \end{bmatrix},$$

From the above representation of $\mathcal{W}$ it follows that

$$\widehat{\mathcal{S}} = -\begin{bmatrix} 0 & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} -I & G \\ S & -I \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 0 \\ Q_\gamma S & Q_\gamma \end{bmatrix},$$

with $Q_\gamma = F(I - SG)^{-1}$. The first factorization in (7.20) follows from the equation $\widehat{\mathcal{S}} = -\mathcal{A}_2\mathcal{W}^{-1}$ and from the fact that $\widehat{\mathcal{S}}$ solves the matrix equation (7.17). Since the roots of $\phi(z)$ are $\xi_1, \ldots, \xi_{m+n}$ and since the eigenvalues of $\mathcal{S}$ are $\xi_1, \ldots, \xi_n$, it follows that the eigenvalues of $\widehat{\mathcal{S}}$ are $\xi_{n+1}^{-1}, \ldots, \xi_{m+n}^{-1}$. Therefore $\rho(\widehat{\mathcal{S}}) < 1$.

The properties of the matrices $\mathcal{T}$ and $\widehat{\mathcal{T}}$ as well as the second factorization of (7.20) can be similarly proved by using the property

$$\begin{bmatrix} E & 0 \\ -H & I \end{bmatrix} \begin{bmatrix} T \\ I \end{bmatrix} = \begin{bmatrix} I & -G \\ 0 & F \end{bmatrix} \begin{bmatrix} T \\ I \end{bmatrix} U_\gamma,$$

with $U_\gamma = \mathcal{C}_\gamma(A - BT)$. □

The solutions S and T can be expressed in functional form by means of the matrix Laurent power series $\psi(s) = \sum_{i=-\infty}^{+\infty} s^i \psi_i = \varphi(s)^{-1}$, where $\varphi(s)$ is the matrix function defined in Theorem 7.5, as shown by the following theorem.

Theorem 7.6 ([BMP10b]). *The constant coefficient ψ_0 of $\psi(s)$ is nonsingular and such that $\psi_0^{-1} = \begin{bmatrix} -I & T \\ S & -I \end{bmatrix}$. Moreover, for the sequence $\{\mathcal{A}_1^{(k)}\}_k$ generated by CR, it holds*

$$\lim_{k\to\infty} \mathcal{A}_1^{(k)} = \lim_{k\to\infty} \begin{bmatrix} -I & G_k \\ H_k & -I \end{bmatrix} = \begin{bmatrix} -I & T \\ S & -I \end{bmatrix}.$$

The convergence is quadratic, and more precisely

$$\|G_k - T\| = O(\nu^{2^k}), \qquad \|H_k - S\| = O(\nu^{2^k}),$$

for any matrix norm $\|\cdot\|$, with

$$\nu = \frac{\max_{i=1,\dots,n} |\xi_i|}{\min_{i=1,\dots,m} |\xi_{n+i}|} < 1.$$

Proof. We observe that the hypotheses of Theorem 5.3 hold since the roots of $\mathcal{A}(s)$ coincide with the eigenvalues ξ_i of $\mathcal{H}_\gamma$ which are split with respect to the unit disk. Moreover, ψ_0 is nonsingular in view of Theorems 5.5 and 7.5.

The first equation in (7.20) can be written as

$$\varphi(s) = \left(I - s \begin{bmatrix} 0 & 0 \\ Q_\gamma S & Q_\gamma \end{bmatrix} \right) \begin{bmatrix} -I & G \\ S & -I \end{bmatrix} \left(I - s^{-1} \begin{bmatrix} R_\gamma & 0 \\ S R_\gamma & 0 \end{bmatrix} \right).$$

For $|s| = 1$, we invert both sides of the last equation to get

$$\psi(s) := \varphi(s)^{-1} = \left(\sum_{j\geqslant 0} s^{-j} \begin{bmatrix} R_\gamma^j & 0 \\ S R_\gamma^j & 0 \end{bmatrix} \right) \mathcal{W}^{-1} \left(\sum_{j\geqslant 0} s^j \begin{bmatrix} 0 & 0 \\ Q_\gamma^j S & Q_\gamma^j \end{bmatrix} \right).$$

The constant term of $\psi(s) = \sum_{i\in\mathbb{Z}} \psi_i s^i$ is

$$\begin{aligned} \psi_0 =& \sum_{j\geqslant 0} \begin{bmatrix} R_\gamma^j & 0 \\ S R_\gamma^j & 0 \end{bmatrix} \mathcal{W}^{-1} \begin{bmatrix} 0 & 0 \\ Q_\gamma^j S & Q_\gamma^j \end{bmatrix} \\ =& \mathcal{W}^{-1} + \begin{bmatrix} I \\ S \end{bmatrix} \left(\sum_{j\geqslant 1} \begin{bmatrix} R_\gamma^j & 0 \end{bmatrix} \mathcal{W}^{-1} \begin{bmatrix} 0 \\ Q_\gamma^j \end{bmatrix} \right) \begin{bmatrix} S & I \end{bmatrix}. \end{aligned} \tag{7.22}$$

From Theorem 5.3, one has

$$\lim_{k\to\infty} \mathcal{A}_1^{(k)} = \psi_0^{-1}$$

that is,

$$\lim_{k\to\infty} \begin{bmatrix} -I & G_k \\ H_k & -I \end{bmatrix} = \psi_0^{-1}. \tag{7.23}$$

Using (7.22), we can say more about the structure of ψ_0^{-1}; defining

$$K = \sum_{j\geqslant 1} \begin{bmatrix} R_\gamma^j & 0 \end{bmatrix} \mathcal{W}^{-1} \begin{bmatrix} 0 \\ Q_\gamma^j \end{bmatrix},$$

we get

$$\psi_0^{-1} = \mathcal{W}^{-1} + \begin{bmatrix} I \\ S \end{bmatrix} K \begin{bmatrix} S & I \end{bmatrix}.$$

Applying the Sherman–Morrison–Woodbury (SMW) formula (2.5) yields

$$\psi_0 = \mathcal{W} + \mathcal{W} \begin{bmatrix} I \\ S \end{bmatrix} \widehat{K}^{-1} \begin{bmatrix} S & I \end{bmatrix} \mathcal{W}, \tag{7.24}$$

where $\widehat{K}$ is the auxiliary matrix to be inverted in the SMW formula. We are not concerned with the explicit value and structure of $\widehat{K}$ now. Since

$$\mathcal{W} \begin{bmatrix} I \\ S \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}, \quad \begin{bmatrix} S & I \end{bmatrix} \mathcal{W} = \begin{bmatrix} 0 & * \end{bmatrix}$$

($*$ stands for an arbitrary block of the right size), the second summand in the right-hand side of (7.24) is nonzero only in its $(1,2)$ block, thus we get

$$\psi_0^{-1} = \begin{bmatrix} -I & * \\ S & -I \end{bmatrix}.$$

Combining the latter equation with (7.23) yields $\lim_{k\to\infty} H_k = S$.

Using the second factorization in (7.20), similarly we may show that

$$\psi_0^{-1} = \begin{bmatrix} -I & T \\ * & -I \end{bmatrix}.$$

Therefore, we conclude that $\lim_{k\to\infty} G_k = T$. □

The quadratic convergence of SDA has been proved in [LX06, GIM07, Guo07] under the more restrictive Assumptions 7.1.1, 7.1.2, 7.1.3, respectively. Our result is valid under the more general Assumption 7.1.

Shrink-and-shift with UL-based transformation

Here, we combine the shrink-and-shift technique with the UL-based transformation in order to arrive at a UQME having a splitting with respect to the unit circle.

Formally, we start from $\widetilde{\mathcal{H}} = I - \frac{1}{t}\mathcal{H}$, factor it as

$$\widetilde{\mathcal{H}} = \begin{bmatrix} D_t & t^{-1}C \\ -t^{-1}B & A_t \end{bmatrix} = \begin{bmatrix} I & -t^{-1}CA_t^{-1} \\ 0 & A_t^{-1} \end{bmatrix}^{-1} \begin{bmatrix} D_t + t^{-2}CA_t^{-1}B & 0 \\ -t^{-1}A_t^{-1}B & I \end{bmatrix},$$

with $D_t := I - t^{-1}D$ and $A_t := I + t^{-1}A$ and reduce it to a UQME with the same structure as (7.5). Cyclic reduction applied to this UQME leads to the same update rule as SDA with initial values

$$\begin{aligned} \widehat{E}_0 &= D_t + t^{-2}CA_t^{-1}B, & \widehat{F}_0 &= A_t^{-1}, \\ \widehat{G}_0 &= t^{-1}CA_t^{-1}, & \widehat{H}_0 &= t^{-1}A_t^{-1}B. \end{aligned} \tag{7.25}$$

The advantage is that we get an algorithm with the same computational cost as SDA that is, $\frac{64}{3}n^3$ ops per step, having somewhat simpler initial values. In the following, we call this algorithm SDA-ss (for shrink-and-shift), and denote the original SDA by SDA-Cayley when it is important to distinguish between the two different sets of initial values rather than focusing on the iteration itself.

The sequences G_k and H_k converge to T and S, respectively; the convergence speed is given by the bounds

$$\|G_k - T\| = O(\nu^{2^k}), \qquad \|H_k - S\| = O(\nu^{2^k})$$

for $\nu = \frac{\max_{i=1,\dots,n}|\mu_i|}{\min_{i=1,\dots,m}|\mu_{n+i}|} < 1$.

Applicability of the SDA-ss algorithm

Theorem 7.7 ([BMP10b]). *Suppose that Assumption 7.1.2 holds with $\mathcal{M}$ irreducible, and choose t as in* (7.11)*. Then for the matrices $\widehat{E}_k, \widehat{F}_k, \widehat{G}_k, \widehat{H}_k$ generated by SDA-ss, i.e., Algorithm 8 with starting values* (7.25)*, one has $\widehat{E}_k, \widehat{F}_k, \widehat{G}_k, \widehat{H}_k \geq 0$, $\widehat{E}_k e > 0$; moreover, $I - \widehat{G}_k\widehat{H}_k$ and $I - \widehat{H}_k\widehat{G}_k$ are nonsingular M-matrices. Therefore, SDA-ss is well-defined.*

Proof. To obtain this proof, we adapt to our case some of the ideas of the convergence and applicability proofs for SDA [GLX06, GIM07].

Let us prove by simultaneous induction the following statements: $\widehat{E}_k, \widehat{F}_k, \widehat{G}_k, \widehat{H}_k \geq 0$; $\widehat{E}_k e > 0$ and $I - \widehat{G}_k\widehat{H}_k$ and $I - \widehat{H}_k\widehat{G}_k$ are nonsingular M-matrices.

First, notice that $A_t^{-1} \geq 0$ since A_t is an M-matrix. Due to the choice of t, $D_t \geq 0$ and thus for the initial values it holds that $\widehat{E}_0, \widehat{F}_0, \widehat{G}_0, \widehat{H}_0 \geq 0$. Moreover, $\widehat{E}_0$ is irreducible since $\widehat{E}_0 = 2I - M$, where M is a Schur complement of $I + t^{-1}\mathcal{M}$, (see e.g. [GIM07, Lemma 2.5]); thus it has no zero rows, from which it follows that $\widehat{E}_0 e > 0$. Moreover, letting $R_t := I - t^{-1}(D - CS)$, one has $R_t \geq 0$. Since $R = D - CS$ is an irreducible M-matrix [Guo01], R_t has no zero rows, thus $R_t e > 0$.

It is proved in Guo *et al.* [GLX06] for the SDA case, but it also holds in our case with the same proof, that

$$S - \widehat{H}_0 = \widehat{F}_0 S R_t, \quad \widehat{E}_0 = (I - \widehat{G}_0 S)R_t.$$

From the first equation it follows that $\widehat{H}_0 \leq S$; from the second, that $(I - \widehat{G}_0 S)R_t e = E_0 e > 0$, and thus $I - \widehat{G}_0 S$ is a nonsingular M-matrix. Thus $I - \widehat{G}_0\widehat{H}_0 \geq I - \widehat{G}_0 S$ is a nonsingular M-matrix, too. By the SMW formula,

$$(I - \widehat{H}_0\widehat{G}_0)^{-1} = I + \widehat{H}_0(I - \widehat{G}_0\widehat{H}_0)^{-1}\widehat{G}_0, \tag{7.26}$$

thus $I - \widehat{H}_0\widehat{G}_0$ is a nonsingular M-matrix.

The inductive step is proved with the same techniques. It is easy to prove that $\widehat{E}_{k+1}, \widehat{F}_{k+1}, \widehat{G}_{k+1}, \widehat{H}_{k+1} \geq 0$; from the fact that $I - \widehat{G}_k\widehat{H}_k$ is nonsingular it follows that $\widehat{E}_{k+1} e > 0$. The two relations

$$S - \widehat{H}_{k+1} = \widehat{F}_{k+1} S R_t^{2^{k+1}}, \quad \widehat{E}_{k+1} = (I - \widehat{G}_{k+1} S)R_t^{2^{k+1}},$$

proved in the same way as above [GLX06], allow one to carry out the same proof as in the base step to conclude that $I - \widehat{G}_{k+1}\widehat{H}_{k+1}$ and $I - \widehat{H}_{k+1}\widehat{G}_{k+1}$ are nonsingular M-matrices. □

Some remarks on stability

For both the original SDA [GIM07] and its shrink-and-shift modification, the matrices to be inverted when computing the initial data ($k = 0$) are "tame" matrices, since they are submatrices or Schur complements of $\mathcal{M} + \gamma I$ (or $\mathcal{M} + t^{-1}I$), whose condition number is controlled by the magnitude of γ (or t).

It is interesting to point out that in view of Theorem 7.7, the computation of one iteration step consists in performing products and additions of nonnegative matrices and in the inversion of the M-matrices $I - G_k H_k$ and $I - H_k G_k$. Multiplication and addition of nonnegative matrices is a numerically stable computation since it does not involve numerical cancellation. The only problematic operation from a stability point of view is the inversion of $I - G_k H_k$ and $I - H_k G_k$. It is proved in C.-H. Guo *et al.* [GIM07] that under Assumption 7.1.2 and 7.2, $I - ST$ is a nonsingular M-matrix, moreover, since G_k and H_k converge monotonically increasing to T and S respectively, one has that $\left\|(I - H_k G_k)^{-1}\right\|_\infty \leq \left\|(I - ST)^{-1}\right\|_\infty$. That is, the norms of the inverses of $I - G_k H_k$ and $I - H_k G_k$ are uniformly bounded. However, when the equation is near to the critical case, $I - ST$ is near to a singular matrix, since in the critical case $I - ST$ is a singular M-matrix. This shows that one could expect some ill-conditioning in the last steps of the algorithm, when H_k and G_k are near to S and T respectively.

This is a problem common to all variants of SDA and all methods based on cyclic reduction; in fact, it depends on the near-singularity of the matrix ψ_0 of Theorem 5.5. This problem is already briefly treated in X.-X. Guo *et al.*[GLX06, Section 5.2], with the conclusion that in practice, when this happens, E_k or F_k are very close to zero (to which they converge quadratically), and thus there is no significant growth in the error bounds.

On the other hand, in view of Theorem 6.11, in the singular case the problem is ill-conditioned, since the best error bound we can obtain is $O(\varepsilon^{1/2})$, with ε being the machine precision. It is therefore to be expected that in a neighborhood of the critical case the approximation error increases accordingly. Up to our knowledge, a complete stability analysis of SDA is not present in the literature yet.

Theoretical comparison of convergence speeds

It is interesting to compare the expected convergence speeds of the two variants of SDA. In view of Theorem 7.6, the convergence ratio is given by

$$\overline{\nu} = \frac{\max_{i=1,\dots,n} |f(\lambda_i)|}{\min_{i=1,\dots,m} |f(\lambda_{n+i})|},$$

with $f(z)$ being either the Cayley transform $\mathcal{C}_\gamma$ or the shrink-and-shift transform f_t. Using the fact that the eigenvalues λ_n and λ_{n+1} are real, one can check that $\max_{i=1,\dots,n} |f(\lambda_i)| = |f(\lambda_n)|$ and $\min_{i=1,\dots,m} |f(\lambda_{n+i})| = |f(\lambda_{n+1})|$. We have

$$\mathcal{C}_\gamma(\lambda) = -\frac{1 - \frac{1}{\gamma}\lambda}{1 + \frac{1}{\gamma}\lambda} = \left(1 - \frac{1}{\gamma}\lambda\right)^2 + O\left(\left(\frac{\lambda}{\gamma}\right)^2\right) = f_\gamma(\lambda)^2 + O\left(\left(\frac{\lambda}{\gamma}\right)^2\right),$$

so when the two ratios $|\lambda_n/\gamma|$ and $|\lambda_{n+1}/\gamma|$ are small, the convergence ratio for the Cayley transform is about the square of the one for the shrink-and-shift with $t = \gamma$. Since the convergence is quadratic, we expect in this case that SDA-ss takes one more iteration than SDA-Cayley to attain the same precision.

It is therefore important to relate the possible choices of γ and t with the two algorithms. Under the Assumptions 7.1.2 and 7.2, when the outermost max in (6.27) is attained by the diagonal of D, or when the two operands are very close, γ for SDA-Cayley and t for SDA-ss can take roughly the same values. Therefore we may expect that SDA-Cayley takes one less iteration. In this case, it is not advisable to adopt SDA-ss, since the computational cost of one more iteration is larger than the difference between the costs of computing the initial values.

n	`sda-Cayley`	`sda-ss`	`ram-ss`
20	1×10^{-14} (21)	1×10^{-14} (22)	4×10^{-15} (22)
100	1×10^{-13} (23)	1×10^{-13} (24)	5×10^{-14} (24)
200	2×10^{-13} (24)	3×10^{-13} (25)	1×10^{-13} (25)
500	1×10^{-12} (25)	1×10^{-12} (26)	4×10^{-13} (27)

Table 7.1: Relative residual (and number of iterations needed) for Test 1, for several choices of the dimension n

On the other hand, when on the diagonal of A there is an element significantly larger than the ones on the diagonal of D, we can choose a value of t much smaller than the best possible for γ. In this case, SDA-ss should be faster than SDA-Cayley, due to both the better convergence ratio and the simpler initial values.

In fact, we can successfully exploit any difference in magnitude between $\max A_{ii}$ and $\max D_{ii}$: when the latter is significantly larger, we may make use of Lemma 6.4 and solve equation (6.7), which swaps $\operatorname{diag}(A)$ and $\operatorname{diag}(D)$, instead. Thanks to this lemma, the minimal solution of (1.5) is the transpose of the minimal solution of (6.7).

7.6 Numerical experiments

We implemented in Matlab and tested the following algorithms:

`sda-Cayley`: the algorithm based on Cayley transform and UL factorization (SDA), as outlined in Section 7.5;

`sda-ss`: the algorithm based on shrink-and-shift and UL factorization, described in Section 7.5;

`ram-ss`: the algorithm based on shrink-and-shift and on Ramaswami's transformation, described in Section 7.5.

For all the algorithms, we reported the minimum value of the residual, computed as

$$\frac{\|XCX + B - AX - XD\|_\infty}{\|XCX + B\|_\infty + \|AX + XD\|_\infty},$$

that the algorithm could achieve, and the number of iterations needed in order to reach it.

We applied the algorithms to the following examples.

Test 1 The structured NARE with coefficients in (1.10). Here we have chosen $\alpha = 10^{-10}$, $c = 1 - 10^{-8}$, which yields an equation very close to the critical case ($\alpha = 0, c = 1$). Several values of the dimension $n = m$ were tested.

Test 2 The equation associated with a randomly chosen singular M-matrix $\mathcal{M}$, generated using the Matlab commands `R=rand(2*n); M=diag(R*ones(2*n,1))-R`. Such matrices are usually very far from the critical case, so the resulting Riccati equation is well conditioned. Several random matrices of dimension $n = 100$ were tested.

Choice	`sda-Cayley`	`sda-ss`	`ram-ss`
1	1×10^{-15} (17)	1×10^{-15} (18)	3×10^{-16} (18)
2	1×10^{-15} (18)	1×10^{-15} (19)	3×10^{-16} (20)
3	8×10^{-16} (13)	7×10^{-16} (14)	2×10^{-16} (14)
4	7×10^{-16} (13)	8×10^{-16} (14)	3×10^{-16} (14)

Table 7.2: Relative residual (and number of iterations needed) for Test 2, for several choices of a random M-matrix of size 200

Problem	`sda-Cayley`	`sda-ss`	`ram-ss`
Example 2	8×10^{-13} (34)	3×10^{-12} (19)	5×10^{-12} (19)
Example 3	6×10^{-12} (18)	3×10^{-12} (4)	2×10^{-12} (4)

Table 7.3: Examples 2 and 3 of Bean *et al.*

Test 3 Examples 2 and 3 of Bean *et al.* [BOT05]: two Riccati equations of small dimension, the former close to the critical case, the latter strongly positive recurrent. These equations are particularly well suited to show how large differences between the magnitude of the diagonals of A and D affect the convergence of the algorithms. Example 2 has $\max A_{ii} = 100.002$, $\max D_{ii} = 0.003$; Example 3 has $\max A_{ii} = 0.018$, $\max D_{ii} = 170.002$ and was solved using the transposed form (6.7).

From the results, it emerges that `ram-ss` is generally able to achieve a marginally better precision, but with a larger computational cost ($\frac{74}{3}n^3$ instead of $\frac{64}{3}n^3$ per iteration). The precisions achieved with the two variants of SDA are usually comparable.

When the diagonal elements of A and D are of similar magnitude, such as in Tests 1 and 2, the analysis of Section 7.5 is confirmed by the experiments: `sda-ss` and `ram-ss` consistently take one more iteration than `sda-Cayley` to achieve the same precision. In these cases, the running time of `sda-ss` (not reported here) is slightly larger, since the simpler starting matrices do not compensate the cost of one more step of the iteration.

On the contrary, in Test 3, with `sda-ss` we have the possibility to choose a much more favorable value of t at no extra cost, due to the difference in magnitude between the diagonals of the coefficients. This provides a significant saving in the number of iterations needed by this algorithm with respect to the other two. In fact, the ratio between the number of iterations needed by `sda-Cayley` and by `sda-ss` is greater than 4, for Example 3.

7.7 Conclusions and research lines

The interpretation provided in this chapter casts new light on SDA and on the relationship between UQMEs and NAREs. With this new setting, several other approaches to the solution of the NARE can be developed. Some possible ideas are:

- using numerical integration and the Cauchy integral theorem for computing the matrix ψ_0 of Theorem 7.6;
- using functional iterations borrowed from stochastic processes (QBD) for solving the UQME;

- using Newton's iteration applied to the UQME trying to exploit the specific matrix structure.

Moreover, it would be interesting to test other functions f mapping $\lambda_1, \ldots, \lambda_n$ inside the unit circle and the other eigenvalues outside, and see if more general results regarding the applicability of the SDA iteration hold true in these cases.

Part II

Rank-structured NAREs

Chapter 8

Storage-optimal algorithms for Cauchy-like matrices

8.1 Introduction

Several classes of algorithms for the numerical solution of Toeplitz-like and Cauchy-like linear systems exist in the literature. We refer the reader to [VBHK01] for an extended introduction on this topic, with descriptions of each method and plenty of citations to the relevant papers, and only summarize them in Table 8.1.

The Levinson algorithm is known to be unstable even for large classes of symmetric positive definite matrices [Cyb80]; stabilization techniques such as *look-ahead* may raise the computational cost from $O(n^2)$ to $O(n^3)$ or from $O(n \log^2 n)$ to $O(n^2)$. A mixed approach like the one in the classical Fortran code by Chan and Hansen [HC92] bounds the complexity growth, but may fail to remove the instability. The GKO algorithm is generally stabler [GKO95], even though in limit cases the growth of the coefficients appearing in the Cauchy-like generators may lead to instability. Though superfast Toeplitz solvers have a lower asymptotic computational cost, when the system matrix is nonsymmetric and ill-conditioned $O(n^2)$ algorithms such as the GKO algorithm [GKO95] are still attractive.

In this chapter we deal with the GKO algorithm for Cauchy-like matrices. Any given Toeplitz matrix can be converted [GKO95] to a Cauchy-like matrix with displacement rank $r = 2$ in $O(n \log n)$ ops and $O(n)$ memory locations, so that algorithms designed for Cauchy-like matrices can deal with Toeplitz and Toeplitz-like systems.

Table 8.1: Existing algorithms for solving Toeplitz and Toeplitz-like linear systems

Name	Operations	Memory	Stability remarks
Levinson	$O(n^2)$	$O(n)$	stable only for some symmetric matrices
Schur-Bareiss	$O(n^2)$	$O(n^2)$	backward stable only for symmetric, positive definite matrices
GKO	$O(n^2)$	$O(n^2)$	stable in practice in most cases
Superfast	$O(n \log^2 n)$	$O(n)$	leading constant may be large; may be unstable in the nonsymmetric case

In 1994, Kailath and Chun [KC94] showed that it is possible to express the solution of a linear system with Cauchy-like matrix as the Schur complement of a certain structured augmented matrix. In 2006, in a paper on Cauchy-like least squares problems, G. Rodriguez exploited this idea to design a variation of GKO using only $O(n)$ memory locations.

In the first part of the present chapter, we provide an alternative $O(n)$-space implementation of the Schur Cauchy-like system solution algorithm, having several desirable computational properties.

Moreover, in some applications, a special kind of partially reconstructible Cauchy-like matrices appears, i.e., those in which the main diagonal is not reconstructible. We call them Trummer-like, as they are associated with Trummer's problem [Ger88]. We show how the $O(n)$-storage algorithms adapt nicely to this case, allowing one to develop an integrated algorithm for their fast inversion. In particular, one of the key steps in order to obtain a full representation of their inverse is the calculation of $\operatorname{diag}(T^{-1})$ for a given Trummer-like T.

8.2 Basic definitions

Throughout the chapter, we say that a vector $s \in \mathbb{C}^n$ is *injective* if $s_i \neq s_j$ for all $i, j = 1, 2, \ldots, n$ such that $i \neq j$.

Displacement operators and Cauchy-like matrices

Let $t, s \in \mathbb{C}^n$. We denote by $\nabla_{t,s}$ the operator $\mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ which maps M to

$$\nabla_{t,s}(M) := \operatorname{diag}(t)M - M\operatorname{diag}(s).$$

A matrix $C \in \mathbb{C}^{n\times n}$ is called *Cauchy-like* (with displacement rank r) if there are vectors t, s and matrices $G \in \mathbb{C}^{n\times r}, B \in \mathbb{C}^{r\times n}$ such that

$$\nabla_{t,s}(C) = GB. \tag{8.1}$$

Notice that if we allow $r = n$, then any matrix is Cauchy-like. In the applications, we are usually interested in cases in which $r \ll n$, since the computational cost of all the involved algorithms depends on r.

A Cauchy-like matrix is called a *quasi-Cauchy matrix* if $r = 1$, and *Cauchy matrix* if $G^* = B = (1, 1, \ldots, 1)$. Usually, it is assumed that the operator $\nabla_{t,s}$ is nonsingular, or equivalently, $t_i \neq s_j$ for all pairs i, j. Under this assumption, the elements of C can be written explicitly as

$$C_{ij} = \frac{\sum_{l=1}^{r} G_{il}B_{lj}}{t_i - s_j}, \tag{8.2}$$

thus C can be fully recovered from G, B, t and s. Otherwise, the latter formula only holds for the entries C_{ij} such that $t_i \neq s_j$, and C is said to be *partially reconstructible*. The matrices G and B are called the *generators* of C, and the elements of t and s are called *nodes*. The vectors t and s are called *node vectors*, or *displacement vectors*.

Trummer-like matrices

In Section 8.6, we deal with the case in which $t = s$ is injective, that is, when the non-reconstructible elements are exactly the ones belonging to the main diagonal. We use ∇_s as a shorthand for $\nabla_{s,s}$. If $\nabla_s(T) = GB$ has rank r, a matrix T is called *Trummer-like* (with displacement rank r). Notice that a Trummer-like matrix can be fully recovered from G, B, s,

and $d = \mathrm{diag}(T)$. Trummer-like matrices are related to interpolation problems [Ger88], and may arise from the transformation of Toeplitz and similar displacement structure [KO97], or directly from the discretization of differential problems [BIP08].

8.3 Overview of the GKO Schur step

Derivation

The fast LU factorization of a Cauchy-like matrix C is based on the following lemma.

Lemma 8.1. *[KS95] Let*

$$C = \begin{bmatrix} C_{1,1} & C_{1,2:n} \\ C_{2:n,1} & C_{2:n,2:n} \end{bmatrix}$$

satisfy the displacement equation (8.1)*, and suppose* $C_{1,1}$ *nonsingular. Then its Schur complement* $C^{(2)} = C_{2:n,2:n} - C_{2:n,1}{C_{1,1}}^{-1}C_{1,2:n}$ *satisfies the displacement equation*

$$\mathrm{diag}(t_{2:n})C^{(2)} - C^{(2)}\,\mathrm{diag}(s_{2:n}) = G^{(2)}B^{(2)},$$

with

$$G^{(2)} = G_{2:n,1:r} - C_{2:n,1}{C_{1,1}}^{-1}G_{1,1:r},\ B^{(2)} = B_{1:r,2:n} - B_{1:r,1}{C_{1,1}}^{-1}C_{1,2:n}. \tag{8.3}$$

Using this lemma, we can construct the LU factorization of C with $O(n^2)$ floating point operations (ops). The algorithm goes on as follows. Given $G^{(1)} = G$, $B^{(1)} = B$, and the two vectors s and t, recover the pivot $C_{1,1}$, the first row $C_{1,2:n}$ and the first column $C_{2:n,1}$ of C using the formula (8.2). This allows to calculate easily the first row of U as $\begin{bmatrix} C_{1,1} & C_{1,2:n} \end{bmatrix}$ and the first column of L as $\begin{bmatrix} 1 & C_{2:n,1}^T{C_{1,1}}^{-1} \end{bmatrix}^T$. Then use equations (8.3) to obtain the generators $G^{(2)}$ and $B^{(2)}$ of the Schur complement $C^{(2)}$ of C. Repeat the algorithm setting $G \leftarrow G^{(2)}$, $B \leftarrow B^{(2)}$, $s \leftarrow s_{2:n}$ and $t \leftarrow t_{2:n}$ to get the second row of U and the second column of L, and so on. A simple implementation is outlined in Algorithm 9. Note that for the sake of clarity we used two different variables L and U; in fact, it is a widely used technique to have them share the same $n \times n$ array, since only the upper triangular part of the matrix is actually used in U, and only the strictly lower triangular part in L. When the

Algorithm 9: LU factorization of Cauchy-like matrices [GKO95]

input: $G \in \mathbb{C}^{n\times r}$, $B \in \mathbb{C}^{r\times n}$, $t, s \in \mathbb{C}^n$ generators of the matrix C
output: LU factors $L, U \in \mathbb{C}^{n\times n}$ of C (can share the same storage space)
  $L \leftarrow I_n$, $U \leftarrow O_n$
  for $k = 1$ to $n-1$ **do**
    $U_{k,\ell} \leftarrow \frac{G_{k,:}B_{:,\ell}}{t_k - s_\ell}$ for all $\ell = k$ to n
    $L_{\ell,k} \leftarrow U_{k,k}^{-1}\frac{G_{\ell,:}B_{:,k}}{t_\ell - s_k}$ for all $\ell = k+1$ to n
    $G_{\ell,:} \leftarrow G_{\ell,:} - L_{\ell,k}G_{k,:}$ for all $\ell = k+1$ to n
    $B_{:,\ell} \leftarrow B_{:,\ell} - U_{k,k}^{-1}B_{:,k}U_{k,\ell}$ for all $\ell = k+1$ to n
  end for
  $U_{n,n} \leftarrow \frac{G_{n,:}B_{:,n}}{t_n - s_n}$
  return L, U

LU factorization is only used for the solution of a linear system in the form $Cx = b$, with

$b \in \mathbb{C}^{n\times m}$, it is a common technique to avoid constructing explicitly L, computing instead $L^{-1}b$ on-the-fly as the successive columns of L are computed. This is also possible with the GKO algorithm, as shown in Algorithm 10.

Algorithm 10: Solving a system $Cx = b$ with implicit L factor and pivoting [GKO95]

input: $G \in \mathbb{C}^{n\times r}$, $B \in \mathbb{C}^{r\times n}$, $t, s \in \mathbb{C}^n$ generators of the matrix C
input: $b \in \mathbb{C}^{n\times m}$ right-hand side
output: $x = C^{-1}b$
  {temporary variables: $l \in \mathbb{C}^n$, $U \in \mathbb{C}^{n\times n}$}
  $U \leftarrow O_n$
  $x \leftarrow b$
  for $k = 1$ to $n-1$ **do**
    $l_\ell \leftarrow \frac{G_{\ell,:}B_{:,k}}{t_\ell - s_k}$ for all $\ell = k$ to n
    $q \leftarrow \text{argmax}_{\ell=k,k+1,\ldots,n} |l_\ell|$ {Finds pivot position}
    $p \leftarrow l_q$ {pivot}
    if p=0 **then**
      print 'error: singular matrix'
    end if
    swap l_k and l_q; $x_{k,:}$ and $x_{q,:}$; $G_{k,:}$ and $G_{q,:}$; t_k and t_q
    $U_{k,k} \leftarrow p$
    $U_{k,\ell} \leftarrow \frac{G_{k,:}B_{:,\ell}}{t_k - s_\ell}$ for all $\ell = k+1$ to n
    $x_{\ell,:} \leftarrow x_{\ell,:} - l_\ell(p^{-1}x_{k,:})$ for all $\ell = k+1$ to n
    $G_{\ell,:} \leftarrow G_{\ell,:} - l_\ell(p^{-1}G_{k,:})$ for all $\ell = k+1$ to n
    $B_{:,\ell} \leftarrow B_{:,\ell} - p^{-1}B_{:,k}U_{k,\ell}$ for all $\ell = k+1$ to n
  end for
  $U_{n,n} \leftarrow \frac{G_{n,:}B_{:,n}}{t_n - s_n}$
  $x_{n,:} \leftarrow x_{n,:}/U_{n,n}$ {start of the back-substitution step}
  for $k = n-1$ down to 1 **do**
    $x_{k,:} \leftarrow x_{k,:} - U_{k,\ell}x_{\ell,:}$ for all $\ell = k+1$ to n
    $x_{k,:} \leftarrow x_{k,:}/U_{k,k}$
  end for
  return x

Comments

Notice that Algorithm 10 includes partial pivoting. Its total cost is $(4r + 2m + 1)n^2 + o(n^2)$ ops, when applied to a matrix C with displacement rank r and an $n \times m$ right-hand side. The algorithm works whenever C is a completely reconstructible Cauchy matrix; if it is not the case, when the number of non-reconstructible entries is small, the algorithm can be modified to store and update them separately, see e.g. Kailath and Olshevsky [KO97] or Section 8.6.

However, there is an important drawback in Algorithm 10: while the size of the input and output data is $O(n)$ (for small values of m and r), $O(n^2)$ memory locations of temporary storage are needed along the algorithm to store U. Therefore, for large values of n the algorithm cannot be effectively implemented on a computer because it does not fit in the RAM.

Moreover, another important issue is caching. Roughly speaking, a personal computer has about 512 kb–8 Mb of cache memory, where the most recently accessed locations of RAM are copied. Accessing a non-cached memory location is an order of magnitude slower than a cached one. The real behavior of a modern processor is more complicated than this simple model, due to the presence of several different levels of cache, each with its own performance, and instruction pipelines [HP03]. Nevertheless, this should highlight that when the used data do not fit anymore into the cache, saving on memory could yield a greater speedup than saving on floating point operations.

8.4 Low-storage version of GKO: the extended matrix approach

Derivation

The following algorithm to solve the high storage issue in GKO was proposed by Rodriguez [Rod06] in 2006, while dealing with least squares Cauchy-like problems. More recently, a deeper analysis and a ready-to-use Matlab implementation were provided by Aricò and Rodriguez [AR09].

The approach is based on an idea that first appeared in Kailath and Chun [KC94]. Let us suppose that C is a completely reconstructible Cauchy-like matrix and that s is injective. The solution of the linear system $Cx = b$ can be expressed as the Schur complement of C in the rectangular matrix

$$\widetilde{C} := \begin{bmatrix} C & b \\ -I & 0 \end{bmatrix}.$$

Thus, we can compute x by doing n steps of Gaussian elimination on $\widetilde{C}$. Moreover, the first block column of $\widetilde{C}$ is a partially reconstructible Cauchy-like matrix with respect to the node vectors

$$\widetilde{s} := \begin{bmatrix} t \\ s \end{bmatrix}$$

and t; therefore, while performing the Gaussian elimination algorithm, the entries of this block can be stored and updated in terms of the generators, as in Algorithm 9. Unlike the previous algorithms, we may discard the rows of U and columns of L as soon as they are computed, keeping only the generators. Instead, the entries in the second block column are computed with customary Gaussian elimination and stored along all the algorithm.

The following observations, which are needed later, should make clearer what is going on with this approach.

Lemma 8.2. *Suppose for simplicity that no pivoting is performed; let L and U be the LU factors of C, x be the solution to the linear system $Cx = b$, y be the solution to $Ly = b$, and $W = U^{-1}$. Let k denote the step of Gaussian elimination being performed, with e.g. $k = 1$ being the step that zeroes out all the elements of the first column but the first. During the algorithm,*

1. *The (i, j) entry of the $(1, 1)$ block is updated at all steps k with $k < \min(i, j+1)$. After its last update, it contains $U_{i,j}$.*
2. *The (i, j) entry of the $(1, 2)$ block is updated at all steps k with $k < i$. After its last update, it contains $y_{i,j}$.*
3. *The (i, j) entry of the $(2, 2)$ block is updated at all steps k with $k \geq i$. In particular, the last step ($k = n$) updates all entries, and after that the $(2, 2)$ block contains $x_{i,j}$.*

4. *The (i, j) entry of the $(2, 1)$ block is updated at all steps k with $i \leq k \leq j$. After its last update, it contains 0. Immediately before that, i.e., just after step $j - 1$, it contains $-W_{i,j}U_{j,j}$.*

Proof. From the structure of Gaussian elimination, it can easily be verified that the entries are only updated during the above mentioned steps. In particular, for the condition on updates to the $(2, 1)$ block, it is essential that the initial $(2, 1)$ block initially contains a diagonal matrix. Regarding which values appear finally in each position,

1. is obvious: in fact, if we ignore all the other blocks, we are doing Gaussian elimination on C.

2. is easily proved: since the row operations we perform transform $C = LU$ to U, they must be equivalent to left multiplication by L^{-1}.

3. is a consequence of the well-known fact that after n steps of Gaussian elimination we get the Schur complement of the initial matrix in the trailing diagonal block.

4. is less obvious. Let us call $Z_{i,k}$ the value of the (i, k) entry of the $(2, 1)$ block right after step $k - 1$, and consider how the entries of the $(2, 2)$ block are updated along the algorithm. They are initially zero, and at the kth step the one in place (i, j) is incremented by $-(Z_{i,k}/U_{k,k})y_{k,j}$, so its final value is

$$x_{i,j} = -\sum_k (Z_{i,k}/U_{k,k})y_{k,j}.$$

Since for each choice of b (and thus of $y = L^{-1}b$) $Z_{i,k}$ and $W_{i,k}$ are unchanged, as they only depend on C, and it holds that

$$x_{i,j} = (U^{-1}y)_{i,j} = \sum_k W_{i,k}y_{k,j},$$

the only possibility is that $W_{i,k} = -Z_{i,k}/U_{k,k}$ for each i, k.

□

We report here the resulting Algorithm 11.

Comments

It is worth mentioning that several nice properties notably simplify the implementation.

- The partial reconstructibility of $\widetilde{C}$ is not an issue. If the original matrix C is fully reconstructible and s is injective, then the non-reconstructible entries of $\widetilde{C}$ are the ones in the form $C(n + k, k)$ for $k = 1, \ldots, n$, that is, the ones in which the -1 entries of the $-I$ block initially lie. It is readily shown that whenever the computation of such entries is required, their value is the initial one of -1.

- At each step of the algorithm, the storage of only n rows of G and of the right block column x is required: at step k, we only need the rows with indices from k to $n + k - 1$ (as the ones below are still untouched by the algorithm, and the ones above are not needed anymore). It is therefore possible to reuse the temporary variables to store the rows modulo n, thus halving the storage space needed for some of the matrices.

Algorithm 11: Solving a system $Cx = b$ with the extended matrix algorithm [AR09]

input: $G \in \mathbb{C}^{n\times r}$, $B \in \mathbb{C}^{r\times n}$, $t, s \in \mathbb{C}^n$ generators of the matrix C
input: $b \in \mathbb{C}^{n\times m}$ right-hand side
output: $x = C^{-1}b$
{temporary variables: $l, u \in \mathbb{C}^n$}
$x \leftarrow b$
for $k = 1$ to $n-1$ **do**
  $l_\ell \leftarrow \frac{G_{\ell,:}B_{:,k}}{s_\ell - s_k}$ for all $\ell = 1$ to $k-1$
  $l_\ell \leftarrow \frac{G_{\ell,:}B_{:,k}}{t_\ell - s_k}$ for all $\ell = k$ to n
  $q \leftarrow \operatorname{argmax}_{\ell=k,k+1,\dots,n} |l_\ell|$ {Finds pivot position}
  $p \leftarrow l_q$ {pivot}
  if p=0 **then**
    print 'error: singular matrix'
  end if
  swap l_k and l_q; $x_{k,:}$ and $x_{q,:}$; $G_{k,:}$ and $G_{q,:}$; t_k and t_q
  $u_\ell \leftarrow \frac{G_{k,:}B_{:,\ell}}{t_k - s_\ell}$ for all $\ell = k+1$ to n
  $x_{k,:} \leftarrow p^{-1}x_{k,:}$
  $x_{\ell,:} \leftarrow x_{\ell,:} - l_\ell x_{k,:}$ for all $\ell \neq k$
  $G_{k,:} \leftarrow p^{-1}G_{k,:}$
  $G_{\ell,:} \leftarrow G_{\ell,:} - l_\ell G_{k,:}$ for all $\ell \neq k$
  $B_{:,\ell} \leftarrow B_{:,\ell} - p^{-1}B_{:,k}u_\ell$ for all $\ell = k+1$ to n
end for
$l_\ell \leftarrow \frac{G_{\ell,:}B_{:,k}}{s_\ell - s_k}$ for all $\ell = 1$ to $n-1$
$p \leftarrow \frac{G_{n,:}B_{:,n}}{t_n - s_n}$
$x_{n,:} \leftarrow p^{-1}x_{n,:}$
$x_{\ell,:} \leftarrow x_{\ell,:} - l_\ell x_{n,:}$ for all $\ell \neq n$
return x

- Pivoting can be easily included without destroying the block structure by acting only on the rows belonging to the first block row of $\widetilde{C}$.

Algorithm 11 uses $(6r + 2m + \frac{3}{2})n^2 + o(n^2)$ floating point operations, and it can be implemented so that the input variables G, B, t, b are overwritten during the algorithm, with x overwriting b, so that it only requires $2n$ memory locations of extra storage (to keep l and u).

As we stated above, for the algorithm to work we need the additional assumption that s is injective, i.e., $s_i \neq s_j$ for all j. This is not restrictive when working with Cauchy-like matrices derived from Toeplitz matrices or from other displacement structured matrices; in fact, in this case the entries s_i are the n complex nth roots of a fixed complex number, thus not only are they different, but their differences $s_i - s_j$ can be easily bounded from below, which is important to improve the stability of the algorithm. This is a common assumption when dealing with Cauchy matrices, since a Cauchy (or quasi-Cauchy) matrix is nonsingular if and only if x and y are injective. For Cauchy-like matrices this does not hold, but the injectivity of the two vectors is still related to the singularity of the matrix: for instance, we have the following result.

Lemma 8.3. *Let s have $r+1$ repeated elements, that is, $s_{i_1} = s_{i_2} = \cdots = s_{i_{r+1}} = s$. Then the Cauchy-like matrix* (8.2) *is singular.*

Proof. Consider the submatrix C' formed by the $r+1$ columns of C with indices $i_1, \ldots, i_{r+1}$. It is the product of the two matrices $G' \in \mathbb{C}^{n\times r}$ and $B' \in \mathbb{C}^{r\times r+1}$, with

$$(G')_{ij} = \frac{G_{ij}}{t_i - s}, \; (B')_{ij} = B_{is_j}.$$

Therefore C' (and thus C) cannot have full rank. □

8.5 Low-storage version of GKO: the downdating approach

Derivation

In this section, we describe a different algorithm to solve a Cauchy-like system using only $O(n)$ locations of memory [Pol10a]. Our plan is to perform the first **for** loop in Algorithm 10 unchanged, thus getting $y = L^{-1}b$, but discarding the computed entries of U which would take $O(n^2)$ memory locations, and then to recover them via additional computations on the generators.

For the upper triangular system $Ux = y$ to be solved incrementally by back-substitution, we would like the entries of the matrix U to be available one row at a time, starting from the last one, and *after* the temporary value $y = L^{-1}b$ has been computed, that is, after the whole LU factorization has been performed.

Let $G^{(k)}$ ($B^{(k)}$) denote the contents of the variable G (resp. B) after step k. The key idea is trying to undo the transformations performed on B step by step, trying to recover $B^{(k)}$ from $B^{(k+1)}$. Because of the way in which the generators are updated in Algorithms 9 and 10, the first row of $G^{(k)}$ and the first column of $B^{(k)}$ are kept in memory untouched by iterations $k+1, \ldots, n$ of the GKO algorithm. Thus we can use them in trying to undo the kth step of Gaussian elimination.

Let us suppose we know $B^{(k+1)}$, i.e., the contents of the second generator B after the $(k+1)$st step of Gaussian elimination, and the values of $G^{(k)}_{k,:}$ and $B^{(k)}_{:,k}$, which are written in G and B by the kth step of Gaussian elimination and afterward unmodified (since the subsequent steps of Algorithm 10 do not use those memory locations anymore).

We start from the second equation of (8.3) and (8.2) for the kth row of U, written using the colon notation for indices.

$$\begin{aligned} B^{(k+1)}_{:,\ell} &= B^{(k)}_{:,\ell} - B^{(k)}_{:,k} {U_{k,k}}^{-1} U_{k,\ell}, & \ell > k, \\ U_{k,\ell} &= \frac{G^{(k)}_{k,:} B^{(k)}_{:,\ell}}{t_k - s_\ell}, & \ell \geq k. \end{aligned}$$

Substituting $B^{(k)}_{:,\ell}$ from the first into the second, and using the $k = \ell$ case of the latter to deal with $U_{k,k}$, we get

$$U_{k,\ell} = \frac{G^{(k)}_{k,:} B^{(k+1)}_{:,\ell}}{t_k - s_\ell} + \frac{G^{(k)}_{k,:} B^{(k)}_{:,k} {U_{k,k}}^{-1}}{t_k - s_\ell} U_{k,\ell} = \frac{G^{(k)}_{k,:} B^{(k+1)}_{:,\ell}}{t_k - s_\ell} + \frac{t_k - s_k}{t_k - s_\ell} U_{k,\ell}$$

and thus

$$U_{k,\ell} = \frac{G^{(k)}_{k,:} B^{(k+1)}_{:,\ell}}{s_k - s_\ell}, \qquad \ell \geq k, \tag{8.4}$$

$$B^{(k)}_{:,\ell} = B^{(k+1)}_{:,\ell} + B^{(k)}_{:,k} {U_{k,k}}^{-1} U_{k,\ell}, \qquad \ell > k. \tag{8.5}$$

The above equations allow one to recover the value of $B^{(k)}_{:,\ell}$ for all $\ell > k$ using only $B^{(k+1)}_{:,\ell}$, $G^{(k)}_{k,:}$ and $B^{(k)}_{:,k}$ as requested.

By applying the method just described repeatedly for $k = n-1, n-2, \ldots, 1$, we are able to recover one at a time the contents of $B^{(n-1)}, B^{(n-2)}, \ldots, B^{(1)}$, which were computed (and then discarded) in the first phase of the algorithm. I.e., at each step we "downdate" B to its previous value, reversing the GKO step. In the meantime, we get at each step $k = n-1, n-2, \ldots, 1$ the kth row of U. In this way, the entries of U are computed in a suitable way to solve the system $Ux = y$ incrementally by back-substitution.

We report here the resulting Algorithm 12.

Algorithm 12: Solving a system $Cx = b$ with the downdating algorithm

input: $G \in \mathbb{C}^{n\times r}$, $B \in \mathbb{C}^{r\times n}$, $t, s \in \mathbb{C}^n$ generators of the matrix C
input: $b \in \mathbb{C}^{n\times m}$ right-hand side
output: $x = C^{-1}b$
  {temporary variables: $l, u \in \mathbb{C}^n$}
  $x \leftarrow b$
  for $k = 1$ to $n-1$ **do**
    $l_\ell \leftarrow \frac{G_{\ell,:}B_{:,k}}{t_\ell - s_k}$ for all $\ell = k$ to n
    $q \leftarrow \text{argmax}_{\ell=k,k+1,\ldots,n} |l_\ell|$ {Finds pivot position}
    $p \leftarrow l_q$ {pivot}
    if p=0 **then**
      print 'error: singular matrix'
    end if
    swap l_k and l_q; $x_{k,:}$ and $x_{q,:}$; $G_{k,:}$ and $G_{q,:}$; t_k and t_q
    $u_k \leftarrow p$
    $u_\ell \leftarrow \frac{G_{k,:}B_{:,\ell}}{t_k - s_\ell}$ for all $\ell = k+1$ to n
    $x_{\ell,:} \leftarrow x_{\ell,:} - p^{-1} l_\ell x_{k,:}$ for all $\ell = k+1$ to n
    $G_{\ell,:} \leftarrow G_{\ell,:} - p^{-1} l_\ell G_{k,:}$ for all $\ell = k+1$ to n
    $B_{:,\ell} \leftarrow B_{:,\ell} - p^{-1} B_{:,k} u_\ell$ for all $\ell = k+1$ to n
  end for
  $u_n \leftarrow \frac{G_{n,:}B_{:,n}}{t_n - s_n}$
  $x_{n,:} \leftarrow x_{n,:}/u_n$ {start of the back-substitution step}
  for $k = n-1$ down to 1 **do**
    $u_\ell \leftarrow \frac{G_{k,:}B_{:,\ell}}{s_k - s_\ell}$ for all $\ell = k+1$ to n
    $B_{:,\ell} \leftarrow B_{:,\ell} + u_k^{-1} B_{:,k} u_\ell$ for all $\ell = k+1$ to n
    $x_{k,:} \leftarrow x_{k,:} - u_\ell x_{\ell,:}$ for $\ell = k+1$ to n
    $x_{k,:} \leftarrow x_{k,:}/u_k$
  end for
  return x

Comments

Notice that pivoting only affects the first phase of the algorithm, since the whole reconstruction stage can be performed on the pivoted version of C without additional row exchanges.

This algorithm has the same computational cost, $(6r + 2m + \frac{3}{2})n^2 + o(n^2)$, and needs the same number of memory locations, $2n$, as the extended matrix approach. Moreover, they

both need the additional property that s be injective, as an $s_k - s_\ell$ denominator appears in (8.4). These facts may lead one to suspect that they are indeed the same algorithm. However, it is to be noted the two algorithms notably differ in the way in which the system $Ux = y$ is solved: in the extended matrix approach we solve this system by accumulating the explicit multiplication $U^{-1}y$, while in the downdating approach we solve it by back-substitution.

Several small favorable details suggest adopting the latter algorithm:

- With the extended matrix approach, we do not get any entry of x before the last step. On the other hand, with the downdating approach, as soon as the first **for** cycle is completed, we get x_n, and then after one step of the downdating part we get x_{n-1}, and so on, getting one new component of the solution at each step. This is useful because in the typical use of this algorithm on Toeplitz matrices, x is the Fourier transform of a "meaningful" vector, such as one representing a signal, or an image, or the solution to an equation. Using the correct ordering, the last entries of a Fourier transform can be used to reconstruct a lower-sampled preview of the original data, with no additional computational overhead, see e.g. Walker[Wal96]. Thus with this approach we can provide an approximate solution after only the first part of the algorithm is completed.
- In the extended matrix version, each step of the algorithm updates $O(nr)$ memory locations. Instead, in the downdating version, for each k, the $(n-k)$th and $(n+k)$th step work on $O(kr)$ memory locations. Therefore, the "innermost" iterations take only a small amount of memory and thus fit better into the processor cache. This is a desirable behavior similar to the one of *cache-oblivious algorithms* [FLPR99].
- In exact arithmetic, at the end of the algorithm the second generator B of the matrix C is reconstructed as it was before the algorithm. In floating point arithmetic, this can be used as an *a posteriori* accuracy test: if one or more entries of the final values of B are not close to their initial value, then there was a noticeable algorithmic error.

8.6 Computations with Trummer-like matrices

A special class of Cauchy-like matrices which may arise in application [Ger88, KO97, BIP08, BMP09] is that of Trummer-like matrices, i.e., those for which the two node vectors coincide ($t = s$) and are injective. The partial reconstructibility of these matrices requires special care to be taken in the implementation of the solution algorithms. The $O(n)$-storage algorithms we have presented can be adapted to deal with this case. Moreover, it is a natural request to ask for an algorithm that computes the generators of T^{-1} given those of T. Such an algorithm involves three different parts: the solution of a linear system with matrix T and multiple right-hand side; the solution of a similar system with matrix T^*, and the computation of $\operatorname{diag}(T^{-1})$. We show that an adaptation of the algorithm presented in Algorithm 8.4 can perform all three at the same time, fully exploiting the fact that these three computations share a large part of the operations involved.

The following results, which are readily proved by expanding the definition of ∇_s on both sides, are simply the adaptation of classical results on displacement ranks (see e.g. Heinig and Rost [HR84]) to the Trummer-like case. Notice the formal similarity with the derivative operator.

Lemma 8.4. *Let $A, B \in \mathbb{C}^{n\times n}$, and let $r(X) = \operatorname{rk} \nabla_s(A)$.*

1. *$\nabla_s(A+B) = \nabla_s(A) + \nabla_s(B)$, so $r(A+B) \le r(A) + r(B)$.*
2. *$\nabla_s(AB) = \nabla_s(A)B + A\nabla_s(B)$, so $r(AB) \le r(A) + r(B)$.*

3. $\nabla_s(A^{-1}) = -A^{-1}\,\nabla_s(A)A^{-1}$, *so* $r(A^{-1}) = r(A)$.

As we saw in Section 8.2, a Trummer-like matrix can be completely reconstructed by knowing only the node vector s, the generators G and B, and its diagonal $d = \operatorname{diag}(T)$. In this section, we are interested in implementing fast—i.e., using $O(n^2)$ ops—and space-efficient—i.e., using $O(n)$ memory locations—matrix-vector and matrix-matrix operations involving Trummer-like matrices stored in this form.

Matrix-vector product

For the matrix-vector product, all we have to do is reconstructing one row at a time of the matrix T and then computing the customary matrix-vector product via the usual formula $(Tv)_i = \sum_j T_{ij}v_j$. Approximate algorithms for the computation of the Trummer-like matrix-vector product with $O(n \log^2 n)$ ops also exist, see e.g. Bini and Pan [BP94b].

Matrix-matrix operations

The matrix product between two Trummer-like matrices T and S is easy to implement: let G_T and B_T (resp. G_S and B_S) be the generators of T (resp. S); then, by Lemma 8.4, the generators of TS are

$$\begin{bmatrix} TG_S & G_T \end{bmatrix}, \quad \begin{bmatrix} B_S \\ B_T S \end{bmatrix},$$

while $\operatorname{diag}(TS)$ can be computed in $O(n^2)$ by recovering at each step one row of T and one column of S and computing their dot product. Sums are similar: the generators of $S+T$ are

$$\begin{bmatrix} G_S & G_T \end{bmatrix}, \quad \begin{bmatrix} B_S \\ B_T \end{bmatrix},$$

and its diagonal is $d_S + d_T$.

Linear systems

Linear system solving is less obvious. Kailath and Olshevsky [KO97] suggested the following algorithm: the GKO Gaussian elimination is performed, but at the same time the computed row $U_{k,k:n}$ and column $L_{k:n,k}$ are used to update the diagonal d to the diagonal of the Schur complement, according to the customary Gaussian elimination formula

$$T^{(k+1)}_{i,i} = T^{(k)}_{i,i} - L_{i,k}(T^{(k)}_{k,k})^{-1}U_{k,i}. \tag{8.6}$$

It is easy to see that this strategy can be adapted to both the extended matrix and the downdating version of the algorithm, thus allowing one to implement GKO with $O(n)$ storage also for this class of matrices.

However, a more delicate issue is pivoting. Kailath and Olshevsky [KO97] do not deal with the general case, since they work with symmetric matrices and with a symmetric kind of pivoting that preserves the diagonal or off-diagonal position of the entries. Let us consider the pivoting operation before the kth step of Gaussian elimination, which consists in choosing an appropriate row q and exchanging the kth and qth rows. The main issue here is that the two non-reconstructible entries that were in position T_{kk} and T_{qq}, now are in positions T_{qk} and T_{kq}. This requires special handling in the construction of the kth row in the Gaussian elimination step, but luckily it does not affect the successive steps of the algorithm, since the kth column and row are not used from step $k+1$ onwards. On the other hand, the

entry T_{qq}, which used to be non-reconstructible before pivoting, is now reconstructible. We may simply ignore this fact, store it in d and update it with the formula (8.6) as if it were not reconstructible. The algorithm is reported here as Algorithm 13. The extended matrix

Algorithm 13: Solving a system $Tx = b$ with the downdating algorithm

input: $G \in \mathbb{C}^{n\times r}$, $B \in \mathbb{C}^{r\times n}$, $s \in \mathbb{C}^n$ generators of the matrix T
input: $d \in \mathbb{C}^n$ diagonal of T
input: $b \in \mathbb{C}^{n\times m}$ right-hand side
output: $x = T^{-1}b$
  {temporary variables: $l, u \in \mathbb{C}^n$}
  {temporary variable: $\sigma \in \mathbb{N}^n$ vector of integer indices used to keep track of the permutation performed during the pivoting}
  $\sigma(i) \leftarrow i$ for all $i = 1$ to n {initializes σ as the identity permutation}
  for $k = 1$ to $n - 1$ **do**
    $l_k \leftarrow d_k$
    $l_\ell \leftarrow \frac{G_{\ell,:}B_{:,k}}{s_{\sigma(\ell)} - s_k}$ for all $\ell = k + 1$ to n
    $q \leftarrow \text{argmax}_{\ell=k,k+1,\ldots,n} |l_\ell|$ {Finds pivot position}
    $p \leftarrow l_q$ {pivot}
    if p=0 **then**
      print 'error: singular matrix'
    end if
    swap l_k and l_q; $x_{k,:}$ and $x_{q,:}$; $G_{k,:}$ and $G_{q,:}$; $\sigma(k)$ and $\sigma(q)$
    $u_k \leftarrow p$
    $u_\ell \leftarrow \frac{G_{k,:}B_{:,\ell}}{s_{\sigma(k)} - s_\ell}$ for all $\ell = k + 1$ to n, $\ell \neq q$
    $u_q \leftarrow d_q$ {non-reconstructible entry that moved off-diagonal after pivoting}
    $x_{\ell,:} \leftarrow x_{\ell,:} - p^{-1} l_\ell x_{k,:}$ for all $\ell = k + 1$ to n
    $G_{\ell,:} \leftarrow G_{\ell,:} - p^{-1} l_\ell G_{k,:}$ for all $\ell = k + 1$ to n
    $B_{:,\ell} \leftarrow B_{:,\ell} - p^{-1} B_{:,k} u_\ell$ for all $\ell = k + 1$ to n
    $d_\ell \leftarrow d_\ell - p^{-1} l_\ell u_\ell$ for all $\ell = k + 1$ to n {Gaussian elimination on the diagonal}
    if $q \neq k$ **then** {d_q may be reconstructible after the pivoting — but we store it explicitly anyway}
      $d_q \leftarrow \frac{G_{q,:}B_{:,q}}{s_{\sigma(q)} - s_q}$
    end if
  end for
  $u_n \leftarrow \frac{G_{n,:}B_{:,n}}{s_{\sigma(n)} - s_n}$
  $x_{n,:} \leftarrow x_{n,:}/u_n$ {start of the back-substitution step}
  for $k = n - 1$ down to 1 **do**
    $u_\ell \leftarrow \frac{G_{k,:}B_{:,\ell}}{s_k - s_\ell}$ for all $\ell = k + 1$ to n
    $B_{:,\ell} \leftarrow B_{:,\ell} + u_k^{-1} B_{:,k} u_\ell$ for all $\ell = k + 1$ to n
    $x_{k,:} \leftarrow x_{k,:} - u_\ell x_{\ell,:}$ for $\ell = k + 1$ to n
    $x_{k,:} \leftarrow x_{k,:}/u_k$
  end for
  return x

version of the algorithm can also be modified in a similar way — we see this in more detail in the next paragraph.

Matrix inversion

Matrix inversion poses an interesting problem too. The generators of T^{-1} can be easily computed as $T^{-1}G$ and $-BT^{-1}$ by resorting to Lemma 13 applied twice on T and T^*. However, whether we try to compute the representation of T^{-1} or directly that of $T^{-1}S$ for another Trummer-like matrix S, we are faced with the problem of computing $\mathrm{diag}(T^{-1})$ given a representation of T. There appears to be no simple direct algorithm to extract it in time $O(n^2)$ from the LU factors of T.

A possible solution could be based on the decomposition $T^{-1} = \mathrm{diag}(f) + F$, with f a vector and F a matrix with $\mathrm{diag}(F) = [0, 0, \ldots, 0]^T$. In fact, notice that F depends only on the generators of the inverse; therefore, after computing them, one could choose any vector v for which $T^{-1}v$ has already been computed (e.g., $G_{:,1}$) and solve for the entries of f in the equation $T^{-1}v = \mathrm{diag}(f)v + Fv$. This solution was attempted in [BMP09], but was found to have unsatisfying numerical properties.

We present here a different solution based on the observations of Lemma 8.2, that allows to compute the diagonal together with the inversion algorithm [Pol10a]. Let us ignore pivoting in this first stage of the discussion. Notice that the last part of Lemma 8.2 shows us a way to compute $(U^{-1})_{1:k,k}$ at the kth step of the extended matrix algorithm. Our plan is to find a similar way to get $(L^{-1})_{k,1:k}$ at the same step, so that we can compute the sums

$$(T^{-1})_{i,i} = \sum_k (U^{-1})_{i,k}(L^{-1})_{k,i}, \quad i = 1, \ldots, n, \tag{8.7}$$

one summand at each step, accumulating the result in a temporary vector.

The following result holds.

Lemma 8.5. *Let T be Trummer-like with generators G and B, nodes s and diagonal d, $T = LU$ be its LU factorization, and $D = \mathrm{diag}(p)$, where $p_i = U_{i,i}$ are the pivots.*

1. *The LU factorization of T^*, the transpose conjugate of T, is $(U^*D^{-1})(DL^*)$.*
2. *The matrix T^* is Trummer-like with nodes s, diagonal d and generators B^* and G^*.*
3. *Let $G^{(k)}$ and $B^{(k)}$ be the content of the variables G and B after the kth step of the GKO algorithm on T, and $\overline{G}^{(k)}$ and $\overline{B}^{(k)}$ be the content of the same variables after the same step of the GKO algorithm on T^*. Then, $\overline{G}^{(k)} = (B^{(k)})^*$ and $\overline{B}^{(k)} = (G^{(k)})^*$*

Proof. The matrices U^*D^{-1} and DL^* are respectively unit lower triangular and upper triangular. Thus the first part holds by the uniqueness of the LU factorization. The second part is clear, and the last one follows by writing down the formula (8.3) for T and T^*. □

Therefore, there is much in common between the GKO algorithm on T and T^*, and the two can be carried on simultaneously saving a great part of the computations involved. Moreover, in the same way as we obtain $(U^{-1})_{1:k,k}$, we may also get at the kth step its equivalent for T^*, i.e., $((DL^*)^{-1})_{1:,k} = p_k(L^{-1})_{k,1:k}$. Since p_k, the kth pivot, is also known, this allows to recover $(L^{-1})_{k,1:k}$.

Thus we have shown a way to recover both $(U^{-1})_{1:k,k}$ and $(L^{-1})_{k,1:k}$ at the kth step of the extended matrix algorithm, and this allows to compute the kth summand of (8.7) for each i.

Pivoting

How does pivoting affect this scheme for the computation of $\mathrm{diag}(T^{-1})$? If $T = PLU$, formula (8.7) becomes

$$(T^{-1})_{i,i} = \sum_k (U^{-1})_{i,k}(L^{-1}P^{-1})_{k,i}, \quad i = 1, \ldots, n. \tag{8.8}$$

The permutation matrix P, of which we already have to keep track during the algorithm, acts on L^{-1} by scrambling the column indices i, so this does not affect our ability to reconstruct the diagonal, as we still have all the entries needed to compute the kth summand at each step k. We only need to take care of the order in which the elements of $(U^{-1})_{1:k,k}$ and $(L^{-1})_{k,1:k}$ are paired in (8.8).

The complete algorithm, which includes pivoting, is reported here as Algorithm 14.

Comments

It is worth noting with the same run of the GKO algorithm we can compute $\mathrm{diag}(T^{-1})$ and solve linear systems with matrices T and T^*, as the two algorithms share many of their computations. In particular, the solutions of the two systems giving $T^{-1}G$ and BT^{-1}, which are the generators of T^{-1}, are computed by the algorithm with no additional effort: the transformations on G and B needed to solve them are exactly the ones that are already performed by the factorization algorithm. Also observe that, since the computation of $(T^{-1})_{i,i}$ spans steps i to n, while d_i is needed from step 1 to step i, we may reuse the vector d to store the diagonal of the inverse. The resulting algorithm has a total computational cost of $(8r + 2m_1 + 2m_2 + 5)n^2$ ops, if we solve at the same time a system $Tx = b$ with $b \in \mathbb{C}^{n\times m_1}$ and a system $yT = c$ with $c \in \mathbb{C}^{m_2\times n}$ (otherwise just set $m_1 = 0$ and/or $m_2 = 0$). The only extra storage space needed is that used for u, l and σ, i.e., the space required to store $2n$ real numbers and n integer indices.

Another observation is that we did not actually make use of the fact that the diagonal of T is non-reconstructible: in principle, this approach works even if C is a Cauchy-like matrix with respect to two different node vectors t and s. This might be useful in cases in which we would rather not compute explicitly the diagonal elements, e.g. because $t_i - s_i$ is very small, and thus would lead to ill-conditioning.

8.7 Numerical experiments

We denote by $\|\cdot\|$ the Euclidean 2-norm for vectors and the Frobenius norm for matrices.

Speed measurements

The speed experiments were performed on a Fortran 90 implementation of the proposed algorithms. The compiler used was the `lf95` Fortran compiler version 6.20c, with command-line options `-o2 -tp4 -lblasmtp4`. The experiments took place on two different computers:

C1 a machine equipped with four Intel® Xeon™ 2.80Ghz CPUs, each equipped with 512kb of L2 cache, and 6 GB of RAM. Since we did not develop a parallel implementation, only one of the processors was actually used for the computations.

C2 a machine equipped with one Intel® Pentium® 4 2.80Ghz CPU with 1024kb of L2 cache, and 512Mb of RAM.

Algorithm 14: Computing the representation of T^{-1} (and solving systems with matrix T and T^*) [Pol10a]

input: $G \in \mathbb{C}^{n\times r}$, $B \in \mathbb{C}^{r\times n}$, $s \in \mathbb{C}^n$ generators of the matrix T
input: $d \in \mathbb{C}^n$ diagonal of T
input: $b \in \mathbb{C}^{n\times m_1}, c \in \mathbb{C}^{m_2\times n}$ for the (optional) solution of $Tx = b$ and $yT = c$
output: $\widetilde{G}$, $\widetilde{B}$, $\widetilde{d}$ generators and diagonal of T^{-1}; optionally, $x = T^{-1}b$, $y = cT^{-1}$
{temporary variables: $l, u \in \mathbb{C}^n$,$\sigma \in \mathbb{N}^n$}
$x \leftarrow b; y \leftarrow c$
$\sigma(i) \leftarrow i$ for all $i = 1$ to n {initializes σ as the identity permutation}
for $k = 1$ to $n-1$ **do**
$l_\ell \leftarrow \frac{G_{\ell,:}B_{:,k}}{s_\ell - s_k}$ for all $\ell = 1$ to $k-1$
$l_k \leftarrow d_k$
$l_\ell \leftarrow \frac{G_{\ell,:}B_{:,k}}{s_{\sigma(\ell)} - s_k}$ for all $\ell = k+1$ to n
$q \leftarrow \text{argmax}_{\ell=k,k+1,\ldots,n} |l_\ell|$ {Finds pivot position}
$p \leftarrow l_q$ {pivot}
if p=0 **then**
print 'error: singular matrix'
end if
swap l_k and l_q; $x_{k,:}$ and $x_{q,:}$; $G_{k,:}$ and $G_{q,:}$; $\sigma(k)$ and $\sigma(q)$
$u_\ell \leftarrow \frac{G_{k,:}B_{:,\ell}}{s_{\sigma(k)} - s_{\sigma(\ell)}}$ for all $\ell = k+1$ to n, $\ell \neq q$ {"extended matrix" computations for T^*}
$u_\ell \leftarrow \frac{G_{k,:}B_{:,\ell}}{s_{\sigma(k)} - s_\ell}$ for all $\ell = k+1$ to n, $\ell \neq q$
$u_q \leftarrow d_q$ {non-reconstructible entry that moved off-diagonal after pivoting}
$x_{k,:} \leftarrow p^{-1}x_{k,:}$; $x_{\ell,:} \leftarrow x_{\ell,:} - l_\ell x_{k,:}$ for all $\ell \neq k$
$G_{k,:} \leftarrow p^{-1}G_{k,:}$; $G_{\ell,:} \leftarrow G_{\ell,:} - l_\ell G_{k,:}$ for all $\ell \neq k$
$B_{:,k} \leftarrow p^{-1}B_{:,k}$; $B_{:,\ell} \leftarrow B_{:,\ell} - B_{:,k}u_\ell$ for all $\ell \neq k$ {the update is performed in the "extended matrix" fashion for B and y too}
$y_{:,k} \leftarrow p^{-1}y_{:,k}$; $y_{:,\ell} \leftarrow y_{:,\ell} - y_{:,k}u_\ell$ for all $\ell \neq k$
$d_\ell \leftarrow d_\ell - p^{-1}l_\ell u_\ell$ for all $\ell = k+1$ to n
if $q \neq k$ **then** {d_q may be reconstructible after the pivoting — but we store it explicitly anyway}
$d_q \leftarrow \frac{G_{q,:}B_{:,q}}{s_{\sigma(q)} - s_q}$
end if
$l_k \leftarrow -1; u_k \leftarrow -1; d_k \leftarrow 0$ {prepares to overwrite $d(k)$ with the diagonal of the inverse}
$u_\ell \leftarrow 0$ for $\ell = k+1$ to n {permutes the entries of u to create the right matching for the update of the formula(8.8). Notice that the variable u holds now the entries of $(L^{-1}P^{-1})_{k,\ell}$ and l holds the entries of $(U^{-1})_{\ell,k}$}
$u_{\sigma(\ell)} \leftarrow u_\ell$ for all $\ell = 1$ to n
$d_\ell \leftarrow d_\ell + p^{-1}l_\ell u_\ell$ for all $\ell = 1$ to k
end for{continues in the next page}

Algorithm 14: (continued) Computing the representation of T^{-1} (and solving systems with matrix T and T^*) [Pol10a]

{continues: last step ($k = n$) of the algorithm}
$l_\ell \leftarrow \frac{G_{\ell,:}B_{:,n}}{s_\ell - s_n}$ for all $\ell = 1$ to $n-1$
$u_\ell \leftarrow \frac{G_{n,:}B_{:,\ell}}{s_{\sigma(n)} - s_{\sigma(\ell)}}$ for all $\ell = 1$ to $n-1$
$p \leftarrow d_n$
$x_{n,:} \leftarrow p^{-1}x_{n,:}$; $x_{\ell,:} \leftarrow x_{\ell,:} - l_\ell x_{n,:}$ for all $\ell = 1$ to $n-1$
$G_{n,:} \leftarrow p^{-1}G_{n,:}$; $G_{\ell,:} \leftarrow G_{\ell,:} - l_\ell G_{n,:}$ for all $\ell = 1$ to $n-1$
$B_{:,n} \leftarrow p^{-1}B_{:,n}$; $B_{:,\ell} \leftarrow B_{:,\ell} - B_{:,n}u_\ell$ for all $\ell = 1$ to $n-1$
$y_{:,n} \leftarrow p^{-1}y_{:,n}$; $y_{:,\ell} \leftarrow y_{:,\ell} - y_{:,n}u_\ell$ for all $\ell = 1$ to $n-1$
$l_n \leftarrow -1; u_n \leftarrow -1; d_n \leftarrow 0$
$u_{\sigma(\ell)} \leftarrow u_\ell$ for all $\ell = 1$ to n
$d_\ell \leftarrow d_\ell + p^{-1}l_\ell u_\ell$ for all $\ell = 1$ to n
$y_{:,\sigma(\ell)} = y_{:,\ell}$ for all $\ell = 1$ to n {undoes the pivoting on the rows of y and B}
$B_{:,\sigma(\ell)} = B_{:,\ell}$ for all $\ell = 1$ to n
return $\widetilde{G} \leftarrow G, \widetilde{B} \leftarrow B, \widetilde{d} \leftarrow d, x, y$

As an indicative comparison with a completely different algorithm, with different stability characteristics, we also reported the computational time for the solution of a (different!) Toeplitz system of the same size with the classical TOMS729 routine from Chan and Hansen, which is a Levinson-based Toeplitz solver. For C1, we reported the times of the Matlab backslash solver as a further comparison.

The results are shown in Table 8.2.

It is clear from the table that two different behaviors arise for different sizes of the input. For small values of n, the winner among the GKO variants is the traditional $O(n^2)$-space algorithm, due to its lower computational cost of $(4r+2m+1)n^2$ instead of $(6r+2m+\frac{3}{2})n^2$ (for these tests, $r = 2$, $m = 1$). As the dimension of the problem increases, cache efficiency starts to matter, and the traditional algorithm becomes slower than its counterparts. This happens starting from $n \approx 256 - 512$. Quick calculations show that the memory occupation of the full $n \times n$ matrix is 512kb for $n = 256$ and 2Mb for $n = 512$, so the transition takes indeed place when the $O(n^2)$ algorithm starts to suffer from cache misses.

The three GKO variants are slower than TOMS729; this is also due to the fact that the implementation of the latter is more mature than the GKO solvers we developed for this test; it uses internally several low-level optimizations such as specialized BLAS routines with loop unrolling.

Accuracy measurements

For the accuracy experiments, we chose four test problems, the first two inspired from Boros, Kailath and Olshevsky [BKO02], the third taken from Gohberg, Kailath and Olshevsky [GKO95], and the fourth from Sweet and Brent [SB95] (with a slight modification).

P1 is a Cauchy-like matrix with $r = 2$, nodes $t_i = a + ib$, $s_j = jb$ for $a = 1$ and $b = 2$, and generators G and B such that $G_{i,1} = 1$, $G_{i,2} = -1$, $B_{1,j} = (-1)^j$, $B_{2,j} = 2$. It is an example of a well-conditioned Cauchy-like matrix; in fact, for $n = 512$, its condition number (estimated with the Matlab function `condest`) is 4E+02, and for $n = 4096$ it is 1.3E+03.

Machine C1					
n	$O(n^2)$-space	Ext. matrix	Downdating	TOMS729	Backslash
128	1.49×10^{-03}	2.02×10^{-03}	2.13×10^{-03}	9.36×10^{-04}	3.44×10^{-03}
256	7.70×10^{-03}	7.68×10^{-03}	8.17×10^{-03}	3.59×10^{-03}	1.69×10^{-02}
512	6.06×10^{-02}	2.92×10^{-02}	3.01×10^{-02}	1.38×10^{-02}	8.41×10^{-02}
1024	2.42×10^{-01}	1.19×10^{-01}	1.24×10^{-01}	5.47×10^{-02}	4.41×10^{-01}
2048	9.62×10^{-01}	4.56×10^{-01}	4.66×10^{-01}	2.56×10^{-01}	$2.61 \times 10^{+00}$
4096	$4.60 \times 10^{+00}$	$1.89 \times 10^{+00}$	$1.91 \times 10^{+00}$	$1.02 \times 10^{+00}$	$1.60 \times 10^{+01}$
8192	$3.04 \times 10^{+01}$	$9.26 \times 10^{+00}$	$8.18 \times 10^{+00}$	$5.41 \times 10^{+00}$	Out of mem.
16384	Out of mem.	$4.06 \times 10^{+01}$	$3.64 \times 10^{+01}$	$2.33 \times 10^{+01}$	Out of mem.
32768	Out of mem.	$1.91 \times 10^{+02}$	$1.64 \times 10^{+02}$	$1.04 \times 10^{+02}$	Out of mem.
65536	Out of mem.	$7.93 \times 10^{+02}$	$7.04 \times 10^{+02}$	$4.17 \times 10^{+02}$	Out of mem.

Machine C2				
n	$O(n^2)$-space	Ext. matrix	Downdating	TOMS729
128	1.58×10^{-03}	2.17×10^{-03}	2.23×10^{-03}	4.50×10^{-04}
256	6.19×10^{-03}	8.10×10^{-03}	8.12×10^{-03}	1.60×10^{-03}
512	6.31×10^{-02}	3.20×10^{-02}	3.15×10^{-02}	5.98×10^{-03}
1024	3.08×10^{-01}	1.27×10^{-01}	1.24×10^{-01}	2.34×10^{-02}
2048	$1.25 \times 10^{+00}$	5.12×10^{-01}	4.94×10^{-01}	9.74×10^{-02}
4096	$4.87 \times 10^{+00}$	$2.04 \times 10^{+00}$	$1.98 \times 10^{+00}$	3.87×10^{-01}
8192	Out of mem.	$8.41 \times 10^{+00}$	$8.05 \times 10^{+00}$	$1.60 \times 10^{+00}$
16384	Out of mem.	$3.72 \times 10^{+01}$	$3.38 \times 10^{+01}$	$7.39 \times 10^{+00}$
32768	Out of mem.	$1.92 \times 10^{+02}$	$1.59 \times 10^{+02}$	$4.83 \times 10^{+01}$
65536	Out of mem.	$9.08 \times 10^{+02}$	$7.99 \times 10^{+02}$	$2.68 \times 10^{+02}$

Table 8.2: Speed experiments: CPU times in seconds for the solution of Cauchy-like/Toeplitz linear systems with the different algorithms

P2 is the same matrix but with $a = 1$ and $b = -0.3$. It is an ill-conditioned Cauchy like matrix; in fact, the condition number estimate is 1E+17 for $n = 512$ and 5E+20 for $n = 4096$.

P3 is the Gaussian Toeplitz matrix [GKO95], i.e., the Toeplitz matrix defined by $T_{i,j} = a^{(i-j)^2}$, with size $n = 512$ and different choices of the parameter $a \in (0, 1)$. It is an interesting test case, since it is a matrix for which the Levinson-based Toeplitz solvers are unstable [GKO95]. Its condition number estimate is 7E+09 for $a = .90$ and 3E+14 for $a = 0.93$.

P4 is a Cauchy-like matrix for which generator growth is expected to occur [SB95]. We chose $n = 128$, the same nodes as **P1**, and generators defined by $G = [a,\ a + \varepsilon f]$, $B = [a + \varepsilon g,\ -a]^T$, where $\varepsilon = 10^{-12}$ and a, f, g are three vectors with random entries uniformly distributed between 0 and 1 (generated with the Matlab `rand` function). Notice that the absolute values of the entries of GB is about 1e-12, and their relative accuracy is about 1e-04.

For **P1** and **P2**, we chose several different values of n, for each of them we computed the product $v = Ce$ (where $e = [11 \dots 1]^T$) with the corresponding matrix C, and applied

the old and new GKO algorithms to solve the system $Cx = v$. We computed the relative error as

$$err = \frac{\|x - e\|}{\|e\|}. \tag{8.9}$$

As a comparison, for **P2** we also reported the accuracy of Matlab's unstructured solver (backslash), which is an $O(n^3)$ algorithm based on Gaussian elimination.

For **P3**, we solved the problem $Tx = v$, with T the Gaussian Toeplitz matrix and $v = Te$, for different values of the parameter a, with several different methods: reduction to Cauchy-like form followed by one of the three GKO-Cauchy solvers presented in this chapter, Matlab's backslash, and the classical Levinson Toeplitz solver TOMS729 by Chan and Hansen [HC92]. The errors reported in the table are computed using the formula (8.9).

For **P4**, we generated five matrices, with the same size and parameters but different choices of the random vectors a, f and g. We used the same right-hand side and error formula as in the experiments **P1** and **P2**. The condition number estimates of the matrices are reported as well.

The results are shown in Table 8.3. There are no significant differences in the accuracy of the three variants of GKO. This shows that, at least in our examples, despite the larger number of operations needed, the space-efficient algorithms are as stable as the original GKO algorithm. On nearly all examples, the stability is on par with that of Matlab's backslash operator. When applied to critical Toeplitz problems, the GKO-based algorithms can achieve better stability results than the classical Levinson solver TOMS729.

In the generator growth case **P4**, the accuracy is very low, as expected from the theoretical bounds; nevertheless, there is no significant difference in the accuracy of the three versions.

We point out that a formal stability proof of the GKO algorithm cannot be established, since it is ultimately based on Gaussian elimination with partial pivoting, for which counterexamples to stability exist, and since in some limit cases there are other issues such as generators growth [SB95]: i.e., the growth of the elements of G and B (but not of L and U) along the algorithm. However, both computational practice and theoretical analysis suggest that the GKO algorithm is in practice a reliable algorithm [Ols03]. Several strategies, such as the one proposed by Gu [Gu98], exist in order to avoid generator growth, and they can be applied to both the original GKO algorithm and its space-efficient versions. The modified versions of GKO are not exempt from this stability problem, since they perform the same operations as the original one plus some others; nevertheless, when performing the numerical experiments, we encountered no case in which the $O(n)$-space algorithms suffer from generator growth while the original version does not.

A posteriori accuracy test

We tested on the experiment **P2** the *a posteriori* accuracy test mentioned at the end of Section 8.5; i.e., solving the system with the downdating approach and then comparing the values of B before and after the algorithm. In Table 8.4, we report the value of the relative error $\|B - B'\| / \|B\|$, where B' is the value of the variable initially holding the second generator B at the end of the algorithm. We compare it with the relative residual $\|C\tilde{x} - b\| / \|b\|$, where C and b are the system matrix and right-hand side of the experiment **P2**, and $\tilde{x}$ is the solution computed by the downdating algorithm. At least in this experiment, the proposed test is not able to capture the instability of the algorithm; the computation of the relative residual is more accurate as an *a posteriori* test to estimate the accuracy of the solution.

Problem P1

n	$O(n^2)$-space	Ext. matrix	Downdating
128	1.26×10^{-15}	1.16×10^{-15}	1.06×10^{-15}
256	1.52×10^{-15}	1.81×10^{-15}	1.46×10^{-15}
512	2.98×10^{-15}	3.06×10^{-15}	3.09×10^{-15}
1024	2.79×10^{-15}	3.43×10^{-15}	3.07×10^{-15}
2048	4.57×10^{-15}	5.92×10^{-15}	5.04×10^{-15}
4096	5.23×10^{-15}	7.45×10^{-15}	5.46×10^{-15}
8192	7.49×10^{-15}	1.25×10^{-14}	7.29×10^{-15}
16384	Out of mem.	1.65×10^{-14}	1.15×10^{-14}
32768	Out of mem.	2.62×10^{-14}	1.76×10^{-14}
65536	Out of mem.	3.93×10^{-14}	2.21×10^{-14}

Problem P2

n	$O(n^2)$-space	Ext. matrix	Downdating	Backslash
128	4.23×10^{-05}	4.23×10^{-05}	4.23×10^{-05}	6.94×10^{-05}
256	2.50×10^{-03}	2.50×10^{-03}	2.50×10^{-03}	1.68×10^{-03}
512	1.31×10^{-01}	1.31×10^{-01}	1.31×10^{-01}	2.15×10^{-01}
1024	$1.63 \times 10^{+01}$	$1.63 \times 10^{+01}$	$1.63 \times 10^{+01}$	$1.87 \times 10^{+01}$
2048	$4.37 \times 10^{+02}$	$4.37 \times 10^{+02}$	$4.37 \times 10^{+02}$	$2.34 \times 10^{+03}$
4096	$2.31 \times 10^{+04}$	$2.31 \times 10^{+04}$	$2.31 \times 10^{+04}$	$1.12 \times 10^{+03}$

Problem P3

a	$O(n^2)$-space	Ext. matrix	Downdating	Backslash	TOMS729
0.85	2.92×10^{-10}	1.58×10^{-10}	1.96×10^{-10}	3.08×10^{-11}	1.63×10^{-10}
0.87	7.08×10^{-10}	6.93×10^{-10}	6.23×10^{-10}	3.67×10^{-10}	2.74×10^{-09}
0.90	1.93×10^{-07}	2.74×10^{-07}	1.81×10^{-07}	1.40×10^{-07}	3.12×10^{-06}
0.91	2.69×10^{-04}	1.65×10^{-04}	2.65×10^{-04}	1.36×10^{-06}	1.21×10^{-04}
0.92	8.09×10^{-05}	1.17×10^{-04}	1.54×10^{-04}	4.64×10^{-05}	1.02×10^{-02}
0.93	5.77×10^{-03}	6.57×10^{-03}	6.18×10^{-03}	2.53×10^{-03}	$2.49 \times 10^{+00}$
0.94	3.77×10^{-01}	$2.11 \times 10^{+00}$	2.84×10^{-01}	$1.12 \times 10^{+00}$	$1.77 \times 10^{+03}$

Problem P4

$O(n^2)$-space	Ext. matrix	Downdating	`condest(C)`
1.68×10^{-01}	1.68×10^{-01}	1.68×10^{-01}	$4 \times 10^{+04}$
1.17×10^{-01}	1.16×10^{-01}	1.13×10^{-01}	$4 \times 10^{+03}$
$2.81 \times 10^{+01}$	$2.80 \times 10^{+01}$	$2.80 \times 10^{+01}$	$1 \times 10^{+05}$
5.55×10^{-02}	5.17×10^{-02}	5.57×10^{-02}	$1 \times 10^{+04}$
6.54×10^{-02}	6.76×10^{-02}	7.39×10^{-02}	$1 \times 10^{+03}$

Table 8.3: Relative forward errors

Problem P2		
n	*A posteriori* test	Relative residual
128	2.67×10^{-12}	2.03×10^{-13}
256	5.61×10^{-12}	4.28×10^{-13}
512	8.83×10^{-12}	1.70×10^{-12}
1024	1.45×10^{-10}	2.94×10^{-09}
2048	6.54×10^{-10}	4.99×10^{-07}
4096	6.25×10^{-10}	3.56×10^{-05}

Table 8.4: Accuracy of the *a posteriori* accuracy test

Problem P2			
n	Downdating(Matlab)	Downdating(Fortran)	Backslash(assembling+solving)
128	4.83×10^{-02}	2.05×10^{-03}	$1.45 \times 10^{-02} + 2.70 \times 10^{-03}$
256	9.22×10^{-02}	7.52×10^{-03}	$1.95 \times 10^{-02} + 1.24 \times 10^{-02}$
512	2.13×10^{-01}	2.78×10^{-02}	$4.02 \times 10^{-02} + 6.62 \times 10^{-02}$
1024	5.81×10^{-01}	1.11×10^{-01}	$1.22 \times 10^{-01} + 3.53 \times 10^{-01}$
2048	$1.85 \times 10^{+00}$	4.33×10^{-01}	$4.52 \times 10^{-01} + 2.22 \times 10^{+00}$
4096	$7.10 \times 10^{+00}$	$1.73 \times 10^{+00}$	$1.91 \times 10^{+00} + 1.43 \times 10^{+01}$

Table 8.5: CPU times (in seconds) on the machine **C1** for the Matlab and Fortran implementation of downdating and for the Matlab backslash operator

Speed comparison with Matlab's backslash

We have compared the speed of the Matlab and Fortran implementations of downdating GKO with the cost of assembling the full matrix C in Matlab and solving the system with the backslash operator. The results are in Table 8.5. The experiments were performed on **C1**, and the version of Matlab used was 7 (R14) SP1.

The comparison is not meant to be fair: on one hand, we are testing a $O(n^2)$ and a $O(n^3)$ algorithm; on the other, we are comparing an interpreted program, a compiled program and a call to a native machine-code library within Matlab. Starting from $n = 2048$, even the Matlab version of the downdating algorithm is faster than backslash: for large values of n, the overhead of processing $O(n)$ instructions with the Matlab interpreter is amortized.

Inversion of Trummer-like matrices

We now turn to testing the algorithm for the structured inversion of Trummer-like matrices proposed in Section 8.6. We chose two experiments, one with well-conditioned matrices and one with ill-conditioned ones. Notice that not all possible choices of the generators G and B are admissible for a Trummer-like matrix, since the displacement equation implies $G_{i,:}B_{:,i} = 0$ for all i.

T1 The $n \times n$ Trummer matrix T with $T_{i,i} = 1$, nodes defined by $s_i = \frac{i}{n}$ and generators defined by $G_{i,1} = i$, $G_{i,2} = -1$, $B_{1,i} = \cos\left(\frac{i\pi}{n}\right)$, $B_{i,2} = G_{i,1}B_{1,i}$ for all $i = 1, \ldots, n$.

Problem T1				
n	E_1	E_2	E_3	`condest(T)`
128	5.45×10^{-16}	1.13×10^{-14}	2.94×10^{-15}	$6 \times 10^{+02}$
256	7.07×10^{-16}	2.85×10^{-14}	7.40×10^{-15}	$1 \times 10^{+03}$
512	9.34×10^{-16}	4.10×10^{-14}	1.17×10^{-14}	$2 \times 10^{+03}$
1024	1.33×10^{-15}	1.28×10^{-13}	2.93×10^{-14}	$5 \times 10^{+03}$
2048	1.87×10^{-15}	2.33×10^{-13}	5.69×10^{-14}	$1 \times 10^{+04}$
4096	2.75×10^{-15}	2.53×10^{-13}	7.94×10^{-14}	$2 \times 10^{+04}$
Problem T2				
ε	E_1	E_2	E_3	
1×10^{-03}	2.27×10^{-11}	5.90×10^{-11}	3.02×10^{-11}	
1×10^{-06}	4.04×10^{-08}	8.09×10^{-08}	4.11×10^{-08}	
1×10^{-09}	4.09×10^{-05}	8.18×10^{-05}	4.13×10^{-05}	
1×10^{-12}	3.26×10^{-02}	6.62×10^{-02}	3.29×10^{-02}	
1×10^{-15}	$1.42 \times 10^{+00}$	$4.82 \times 10^{+00}$	$1.73 \times 10^{+00}$	

Table 8.6: Accuracy of the algorithm for inverting Trummer-like matrices

T2 The 512×512 diagonal-plus-rank-1 matrix depending on a parameter ε and defined by

$$T = (1+\varepsilon)I - uu^T, \qquad u = \frac{v}{\|v\|}, \qquad v_i = \frac{i}{n} \text{ for all } i = 1, \ldots, n.$$

Its inverse can be computed explicitly as $(1+\varepsilon)^{-1}(I + \varepsilon^{-1}uu^T)$, and its condition number is $\varepsilon^{-1} + 1$. A diagonal-plus-rank-1 matrix is Trummer-like with $r = 2$ with respect to any set of nodes; in this experiment, we used the node vector defined by $s_i = a + ib$ for all $i = 1, \ldots, n$, with $a = 1$, $b = -0.3$.

We computed the relative errors

$$E_1 := \frac{\|d' - d''\|}{\|d''\|}, \qquad E_2 := \frac{\|G' - G''\|}{\|G''\|} + \frac{\|B' - B''\|}{\|B''\|}, \qquad E_3 := \frac{\|T' - T''\|}{\|T''\|},$$

where G', B', d', T' are the generators, diagonal and full inverse computed by Algorithm 14, and G'', B'', d'', T'' are their reference values computed with Matlab's function `inv` for **T1** and with the exact formula for the inverse for **T2**. The results are reported in Table 8.6. In both cases, the algorithm is able to reach good accuracy, compatibly with the restrictions imposed by the condition number of the matrices.

The Fortran and Matlab implementations of the algorithms that were used in the experiments are available online on `http://arxiv.org/e-print/0903.4569` along with the e-print of the paper on which this chapter is based.

8.8 Conclusions and research lines

In this chapter, we proposed a new $O(n)$-space version of the GKO algorithm for the solution of Cauchy-like linear systems [Pol10a]. Despite the slightly larger number of operations needed, this algorithm succeeds in making a better use of the internal cache memory of

the processor, thus providing an improvement with respect to both the customary GKO algorithm and a similar $O(n)$-space algorithm proposed by Rodriguez [Rod06], [AR09]. Starting from $n \approx 500 - 1000$, the algorithm outperforms these two versions of GKO. When applying this algorithm to the special case of inversion of Trummer-like matrices, several small optimizations reduce the total number of operations needed.

The $O(n)$-space implementation is a trade-off between the number of operations needed and the memory access, which starts to pay off when the matrices are large enough. It is an interesting question whether a $O(n)$-space implementation can be obtained with a lower computational cost, hopefully matching the one of the traditional GKO algorithm.

Another open issue is the possibility to perform an adaptive algorithm, which uses the $O(n)$ approach in a first phase, until the problem is small enough to fit in the cache, and then switches to the traditional GKO implementation.

Chapter 9

Newton method for rank-structured algebraic Riccati equations

9.1 The neutron transport equation

Equation (1.5) with coefficients given by (1.10) arises from the discretization of an integrodifferential equation appearing in a neutron transport model [JL99]. The solution of interest is the minimal positive one, whose existence is proved in [JL99]. Note that for this equation $\mathcal{M}$ is an M-matrix [Guo01]: in fact,

$$\mathcal{M} = \begin{bmatrix} \Gamma & 0 \\ 0 & \Delta \end{bmatrix} - \begin{bmatrix} q \\ e \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix};$$

by Lemma 2.6 this is an M-matrix if and only if

$$0 \leq 1 - \begin{bmatrix} e^T & q^T \end{bmatrix} \begin{bmatrix} \Gamma^{-1} & 0 \\ 0 & \Delta^{-1} \end{bmatrix} \begin{bmatrix} q \\ e \end{bmatrix},$$

which reduces to

$$1 \geqslant e^T \Gamma^{-1} q + q^T \Delta^{-1} e = \sum_{i=1}^{n} \frac{c(1-\alpha)}{2} w_i + \sum_{i=1}^{n} \frac{c(1+\alpha)}{2} w_i = c, \tag{9.1}$$

in view of (1.11). According to the terminology in (6.2), the equation is nonsingular whenever $c \neq 1$, transient when $\alpha \neq 0, c = 1$, and null recurrent when $\alpha = 0, c = 1$.

As this equation is a special case of (1.5), we may apply the methods that were introduced in the previous chapters, such as functional iterations, Newton's method, or SDA. All these algorithms share the same order of complexity, that is, $O(n^3)$ arithmetic operations (ops) per step, and provide quadratic convergence in the noncritical case.

However, observing that the coefficients in (1.10) are defined by a linear number of parameters, it is quite natural to aim to design algorithms which exploit this structure to get a lower computational cost.

A step in this direction has been done by L.-Z. Lu [Lu05b]. If we use (1.10) and set $\delta = \operatorname{diag}(\Delta)$, $\gamma = \operatorname{diag}(\Gamma)$, then we can rewrite (1.5) as

$$\nabla_{\delta,-\gamma}(X) = X\Gamma + \Delta X = (Xq + e)(e^T + q^T X); \tag{9.2}$$

therefore any solution X is Cauchy-like with respect to $(\Delta, -\Gamma)$ and its generators are given by

$$u := Xq + e, \qquad v^T := e^T + q^T X. \tag{9.3}$$

We may eliminate X by combining (9.2) and (9.3); the resulting equation can be rewritten as (1.12), with

$$P_{ij} := \frac{q_j}{\delta_i + \gamma_j}, \qquad \tilde{P} := \frac{q_j}{\gamma_i + \delta_j}.$$

Lu proposed first a functional iteration [Lu05b] and then Newton's method [Lu05a] to solve (1.12). Since the equation falls in the framework exposed in Chapter 3, the two methods are well-defined and their convergence is monotonic. The functional iteration has cost $O(n^2)$ but converges linearly, which is a severe drawback since the cases of interest are close-to-critical, and thus the convergence is very slow. On the other hand, Newton's method can be implemented in $O(n^2)$ using the results of Chapter 8 [BIP08] and is very efficient.

In this chapter we show that the Newton method for the NARE (1.10) can be implemented with cost $O(n^2)$ as well, exploiting the structural properties of the matrices to invert. Moreover, the Newton method for (1.12) is linked by simple algebraic relations to the one for the original NARE.

In the null recurrent case, where the Jacobian at the solution is singular, the convergence of Newton's (and therefore Lu's) iteration turns to linear; the mixed iteration proposed in [Lu05a] loses its quadratic convergence, too, while the iteration of [Lu05b] converges sublinearly. Moreover, in view of Theorem 6.11, we may only expect an accuracy of the order of the square root of the machine precision, unless we exploit the singularity of the equation in our algorithm.

We show how we can remove the singularity of the Jacobian and consequently of all the above mentioned drawbacks. The idea is to apply the shift technique originally introduced by C. He, B. Meini and N.H. Rhee [HMR02] and used in the framework of Riccati equations by C.-H. Guo, B. Iannazzo, and B. Meini in [GIM07] and by C.-H. Guo [Guo06]. With this technique, we replace the original Riccati equation with a new one having the same minimal solution but nonsingular Jacobian. We prove that the matrix coefficients of the new equation share the same rank structure properties of the original coefficients in (1.10). This enables us to design a fast Newton iteration which preserves the quadratic convergence and keeps the same $O(n^2)$ complexity even in the critical case.

Moreover, with the shift technique, the information on the singularity of $\mathcal{M}$ is plugged into the algorithm and we may achieve full accuracy in the approximation as confirmed by the numerical experiments.

9.2 Newton's method

Newton's iteration applied to (1.5), for a suitable initial value $X^{(0)}$, generates the matrix sequence $\{X^{(k)}\}$ defined by the solution of the Sylvester equation [GL00]

$$(X^{(k+1)} - X^{(k)})(D - CX^{(k)}) + (A - X^{(k)}C)(X^{(k+1)} - X^{(k)}) = \mathfrak{R}(X^{(k)}), \tag{9.4}$$

where $\mathfrak{R}(X)$ is as in (6.8). Using the Kronecker product notation, this can be written as

$$\operatorname{vec} X^{(k+1)} - \operatorname{vec} X^{(k)} = \left((D - CX^{(k)})^T \otimes I_n + I_n \otimes (A - X^{(k)}C)\right)^{-1} \operatorname{vec} \mathfrak{R}(X^{(k)}), \tag{9.5}$$

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 $M_{X^{(k)}} = (D - CX^{(k)})^T \otimes I_n + I_n \otimes (A - X^{(k)}C)$ is nonsingular for each k. With abuse of notation, we call the matrix M_X the *Jacobian matrix* at X; in fact it is the Jacobian matrix of the vector function $-\operatorname{vec} \circ \mathfrak{R} \circ \operatorname{vec}^{-1}$ at $\operatorname{vec}(X)$.

In the following, we consider a slightly more general case; i.e., when the matrix M is a generic diagonal plus rank-one matrix. Hence, we replace (1.10) with

$$A = \Delta - \widetilde{e}q^T, \quad B = \widetilde{e}e^T, \quad C = \widetilde{q}q^T, \quad D = \Gamma - \widetilde{q}e^T, \tag{9.6}$$

where e, q, $\widetilde{e}$, $\widetilde{q}$ are any nonnegative vectors such that $\mathcal{M}$, as defined in (1.6), is a nonsingular M-matrix or a singular irreducible M-matrix. Such generalization is useful when dealing with the shifted case.

Observe that Newton's iteration for this case can be rewritten as

$$\begin{aligned} X^{(k+1)}\Gamma + \Delta X^{(k+1)} = &-(X^{(k)}\widetilde{q} - X^{(k+1)}\widetilde{q})(q^T X^{(k)} - q^T X^{(k+1)}) \\ &+ (X^{(k+1)}\widetilde{q} + \widetilde{e})(e^T + q^T X^{(k+1)}), \end{aligned} \tag{9.7}$$

and this shows that $X^{(k+1)}$ is Cauchy-like with displacement rank 2. The property holds for all the iterates $X^{(k)}$, $k \geqslant 1$ of Newton's method obtained with any starting matrix $X^{(0)}$.

The Jacobian at $X^{(k)}$, in Kronecker product notation, takes the form

$$M_{X^{(k)}} = \Gamma^T \otimes I_n + I_n \otimes \Delta - (e + X^{(k)T}q)\widetilde{q}^T \otimes I_n - I_n \otimes (\widetilde{e} + X^{(k)}\widetilde{q})q^T.$$

By setting $\mathcal{D} = \Gamma^T \otimes I_n + I_n \otimes \Delta$, $u^{(k)} = \widetilde{e} + X^{(k)}\widetilde{q}$, $v^{(k)} = e + X^{(k)T}q$, and

$$U^{(k)} = \begin{bmatrix} v^{(k)} \otimes I_n & I_n \otimes u^{(k)} \end{bmatrix}, \qquad V = \begin{bmatrix} \widetilde{q}^T \otimes I_n \\ I_n \otimes q^T \end{bmatrix}, \tag{9.8}$$

we can rewrite $M_{X^{(k)}}$ as

$$M_{X^{(k)}} = \mathcal{D} - U^{(k)}V. \tag{9.9}$$

Since $U^{(k)} \in \mathbb{R}^{n^2 \times 2n}$ and $V \in \mathbb{R}^{2n \times n^2}$, the inversion of $M_{X^{(k)}}$ can be reduced to the inversion of a $2n \times 2n$ matrix using the SMW formula:

$$M_{X^{(k)}}^{-1} = \mathcal{D}^{-1} + \mathcal{D}^{-1}U^{(k)}(I_{2n} - V\mathcal{D}^{-1}U^{(k)})^{-1}V\mathcal{D}^{-1}. \tag{9.10}$$

As we show in the next section, this $2n \times 2n$ matrix is Trummer-like, and this fact can be exploited in the computation. This provides an algorithm which implements the Newton method using only $O(n^2)$ ops per step; we report it as Algorithm 15.

Note that the matrix–vector product $\mathcal{D}^{-1}\operatorname{vec} W$ can be reshaped to $\operatorname{vec}^{-1}(\Delta^{-1}W + W\Gamma^{-1})$ and thus implemented with $O(n^2)$ ops. Similarly, the identities

$$\begin{aligned} (v^T \otimes I_n)\operatorname{vec}(W) &= Wv, \\ (I_n \otimes v^T)\operatorname{vec}(W) &= W^T v \end{aligned}$$

allow one to compute the products with $U^{(k)}$ and V in $O(n^2)$ as well. Therefore the overall cost of Algorithm 15 is $O(n^2)$.

9.3 Fast Gaussian elimination for Cauchy-like matrices

We now address the problem of solving the linear system with matrix $R^{(k)}$, given the vector b and the vectors $q, u^{(k)}, v^{(k)}$ such that

$$R^{(k)} = I_{2n} - \begin{bmatrix} \widetilde{q}^T \otimes I_n \\ I_n \otimes q^T \end{bmatrix} \mathcal{D}^{-1} \begin{bmatrix} v^{(k)} \otimes I_n & I_n \otimes u^{(k)} \end{bmatrix}. \tag{9.11}$$

Algorithm 15: Fast Newton's method for the NARE (1.10)

input: $\Delta, \Gamma, e, q, \widetilde{e}, \widetilde{q}$ defining an instance of (9.6)
output: its minimal solution X

$X_0 \leftarrow 0$
$k \leftarrow 0$
while a suitable stopping criterion is not satisfied **do**
  $u^{(k)} \leftarrow \widetilde{e} + X^{(k)}\widetilde{q}$
  $v^{(k)} \leftarrow e + X^{(k)T}q$
  $R \leftarrow \mathfrak{R}(X^{(k)}) = u^{(k)}v^{(k)T} - X^{(k)}\Gamma - \Delta X^{(k)}$
  $b \leftarrow V(\mathcal{D}^{-1}\operatorname{vec}(R))$
  Solve $(I_{2n} - V\mathcal{D}^{-1}U^{(k)})x = b$ using Algorithm 13 — see Section 9.3
  $X^{(k+1)} \leftarrow \mathcal{D}^{-1}\operatorname{vec}(R) + U^{(k)}x$
  $k \leftarrow k + 1$
end while
return $X^{(k)}$

First note that $R^{(k)}$ is a nonsingular M-matrix by Lemma 2.6 applied to the nonsingular M-matrix $M_{X^{(k)}}$ of (9.9). Carrying out the products in (9.11) yields

$$R^{(k)} = I_{2n} - \begin{bmatrix} G^{(k)} & H^{(k)} \\ K^{(k)} & L^{(k)} \end{bmatrix} \tag{9.12}$$

with

$$\begin{aligned} G^{(k)} &= \operatorname{diag}(g_i^{(k)}), & g_i^{(k)} &= \sum_{l=1}^{n} \frac{v_l^{(k)}\widetilde{q}_l}{d_l + \delta_i}, \\ H^{(k)} &= (h_{ij}^{(k)}), & h_{ij}^{(k)} &= \frac{u_i^{(k)}\widetilde{q}_j}{d_j + \delta_i}, \\ K^{(k)} &= (\kappa_{ij}^{(k)}), & \kappa_{ij}^{(k)} &= \frac{v_i^{(k)}q_j}{d_i + \delta_j}, \\ L^{(k)} &= \operatorname{diag}(l_i^{(k)}), & l_i^{(k)} &= \sum_{l=1}^{n} \frac{u_l^{(k)}q_l}{d_i + \delta_l}. \end{aligned} \tag{9.13}$$

Thus $G^{(k)}$ and $L^{(k)}$ are diagonal, and $H^{(k)}$ and $K^{(k)}$ are Cauchy-like. Their displacement equations are

$$\Delta H^{(k)} + H^{(k)}D = u^{(k)}\widetilde{q}^T, \quad DK^{(k)} + K^{(k)}\Delta = v^{(k)}q^T.$$

Partition x and b according to the block structure of $R^{(k)}$ as $x = [x_1^T, x_2^T]^T$, $b = [b_1^T, b_2^T]^T$. Performing the block LU factorization of $R^{(k)}$ enables one to rewrite the system $R^{(k)}x = b$ as

$$\begin{bmatrix} I - G^{(k)} & -H^{(k)} \\ 0 & S^{(k)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ \widehat{b}_2 \end{bmatrix} \tag{9.14}$$

where $S^{(k)} = I - L^{(k)} - K^{(k)}(I - G^{(k)})^{-1}H^{(k)}$ and $\widehat{b}_2 = b_2 - K^{(k)}(I - G^{(k)})^{-1}b_1$. The matrices $I - G^{(k)}$ and $S^{(k)}$ are nonsingular as they are a principal submatrix and the Schur complement of a nonsingular M-matrix, respectively. Moreover, $S^{(k)}$ enjoys the following displacement structure

$$DS^{(k)} - S^{(k)}D = K^{(k)}(I - G^{(k)})^{-1}u^{(k)}\widetilde{q}^T - v^{(k)}q^T(I - G^{(k)})^{-1}H^{(k)}.$$

This can be easily proved since D, Δ, $I - G^{(k)}$, $I - L^{(k)}$ all commute because they are diagonal, in fact,

$$\begin{aligned} DS^{(k)} &= D(I - L^{(k)}) - DK^{(k)}(I - G^{(k)})^{-1}H^{(k)} \\ &= (I - L^{(k)})D + (K^{(k)}\Delta - v^{(k)}q^T)(I - G^{(k)})^{-1}H^{(k)} \\ &= (I - L^{(k)})D + K^{(k)}(I - G^{(k)})^{-1}\Delta H^{(k)} - v^{(k)}q^T(I - G^{(k)})^{-1}H^{(k)} \\ &= (I - L^{(k)})D - K^{(k)}(I - G^{(k)})^{-1}(H^{(k)}D - u^{(k)}\widetilde{q}^T) - v^{(k)}q^T(I - G^{(k)})^{-1}H^{(k)} \\ &= S^{(k)}D + K^{(k)}(I - G^{(k)})^{-1}u^{(k)}\widetilde{q}^T - v^{(k)}q^T(I - G^{(k)})^{-1}H^{(k)}. \end{aligned}$$

Thus $S^{(k)}$ is Trummer-like with displacement rank 2 with respect to the singular operator $DS^{(k)} - S^{(k)}D$. Therefore, we can solve for x_2 in (9.14) using Algorithm 13; once x_2 is recovered, we may get x_1 by substituting it directly in the equation given by the first block row of (9.14).

9.4 Lu's iteration

L.-Z. Lu [Lu05a] proposed to solve the equation (1.5) when the coefficients are in the form (9.6) by applying Newton's iteration to the equivalent equation (1.12). Applicability and monotonic convergence of this iteration follow from the more general arguments in Chapter 3. If one writes down (3.6) for this equation, the algorithm can be expressed as the following iteration for the sequences $\{\widehat{u}^{(k)}\}$, $\{\widehat{v}^{(k)}\}$, $k \geqslant -1$:

$$\begin{bmatrix} \widehat{u}^{(k+1)} \\ \widehat{v}^{(k+1)} \end{bmatrix} = (\widehat{R}^{(k)})^{-1} \begin{bmatrix} \widetilde{e} - \widehat{H}^{(k)}\widehat{v}^{(k)} \\ e - \widehat{K}^{(k)}\widehat{u}^{(k)} \end{bmatrix}, \tag{9.15}$$

starting from $\widehat{u}^{(-1)} = \widehat{v}^{(-1)} = 0$. As we see later on, indexing from $k = -1$ simplifies the subsequent analysis. Here $\widehat{R}^{(k)}$, $\widehat{H}^{(k)}$ and $\widehat{K}^{(k)}$ are defined as

$$\widehat{R}^{(k)} = I_{2n} - \begin{bmatrix} \widehat{G}^{(k)} & \widehat{H}^{(k)} \\ \widehat{K}^{(k)} & \widehat{L}^{(k)} \end{bmatrix}, \tag{9.16}$$

$$\begin{aligned} \widehat{G}^{(k)} &= \operatorname{diag}(\widehat{g}_i^{(k)}), \quad & \widehat{g}_i^{(k)} &= \sum_{l=1}^{n} \frac{\widehat{v}_l^{(k)}\widetilde{q}_l}{d_l + \delta_i}, \\ \widehat{H}^{(k)} &= (\widehat{h}_{ij}^{(k)}), & \widehat{h}_{ij}^{(k)} &= \frac{\widehat{u}_i^{(k)}\widetilde{q}_j}{d_j + \delta_i}, \\ \widehat{K}^{(k)} &= (\widehat{\kappa}_{ij}^{(k)}), & \widehat{\kappa}_{ij}^{(k)} &= \frac{\widehat{v}_i^{(k)}q_j}{d_i + \delta_j}, \\ \widehat{L}^{(k)} &= \operatorname{diag}(\widehat{l}_i^{(k)}), & \widehat{l}_i^{(k)} &= \sum_{l=1}^{n} \frac{\widehat{u}_l^{(k)}q_l}{d_i + \delta_l}, \end{aligned} \tag{9.17}$$

which are, formally, the same relations as in (9.12) and (9.13).

As a first result, since both algorithms are based on the solution of a system with the same structure, we obtain that Algorithm 13 can also be used in the implementation of Lu's iteration to reduce its computational cost to $O(n^2)$. But there is a deeper connection between the two algorithms.

Theorem 9.1 ([BIP08])**.** *Let $\{\widehat{u}^{(k)}\}$, $\{\widehat{v}^{(k)}\}$, $k \geqslant -1$ be the sequences generated by Lu's algorithm for the NARE* (1.5) *with* (9.6)*, and let $\{X^{(k)}\}$, $k \geqslant 0$ be the sequence generated by Newton's iteration with starting point $X^{(0)} = 0$ for the same problem. Then, for all $k \geqslant 0$, one has*

$$\widehat{u}^{(k)} = X^{(k)}\widetilde{q} + \widetilde{e}, \qquad \widehat{v}^{(k)} = X^{(k)T}q + e.$$

Proof. We prove the result by induction over k. It is easy to check from the definitions that $\widehat{R}^{(-1)} = I_{2n}$, and thus $\widehat{u}^{(0)} = \widetilde{e}$, $\widehat{v}^{(0)} = e$; therefore the base step $k = 0$ holds. As a side note, this means that we can save an iteration by starting the computation from $u^{(0)} = \widetilde{e}$, $v^{(0)} = e$.

Assuming by induction that $\widehat{u}^{(k)} = X^{(k)}\widetilde{q} + \widetilde{e} = u^{(k)}$, $\widehat{v}^{(k)} = X^{(k)T}q + e = v^{(k)}$, we find that equations (9.13) and (9.16) define the same matrices; therefore, from now on, we drop the superscript (k) and the hat symbol to ease the notation.

We have

$$V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X) = V\mathcal{D}^{-1}\operatorname{vec}(uv^T) - V\operatorname{vec}X = \begin{bmatrix}\widetilde{e} - (I-G)u\\ e - (I-L)v\end{bmatrix},$$

in view of the relations

$$\mathcal{D}^{-1}\operatorname{vec}(X\Gamma + \Delta X) = \operatorname{vec}X,$$
$$V\mathcal{D}^{-1}\operatorname{vec}(uv^T) = \begin{bmatrix}Gu\\ Lv\end{bmatrix},$$

which can be easily verified from the definitions of $\mathcal{D}$, G and L, where V is the matrix defined in (9.8).

Applying the operator V to both sides of (9.5) yields

$$\begin{aligned} V\operatorname{vec}(X^{(k+1)} - X) =& V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X) + V\mathcal{D}^{-1}U(I_{2n} - V\mathcal{D}^{-1}U)^{-1}V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X)\\ =& (I_{2n} + V\mathcal{D}^{-1}U(I_{2n} - V\mathcal{D}^{-1}U)^{-1})V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X)\\ =& (I_{2n} - V\mathcal{D}^{-1}U)^{-1}V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X), \end{aligned} \tag{9.18}$$

where the last equation holds since $I + M(I-M)^{-1} = (I-M)^{-1}$.

We recall that $R = (I_{2n} - V\mathcal{D}^{-1}U)$ and

$$\begin{bmatrix}\widehat{u}^{(k+1)}\\ \widehat{v}^{(k+1)}\end{bmatrix} = R^{-1}\begin{bmatrix}\widetilde{e} - Hu\\ e - Kv\end{bmatrix}$$

(the latter one being Lu's iteration). Now we can explicitly compute

$$\begin{aligned} R\begin{bmatrix}\widehat{u}^{k+1} - u\\ \widehat{v}^{k+1} - v\end{bmatrix} =& \begin{bmatrix}\widetilde{e} - Hu\\ e - Kv\end{bmatrix} - \left(I_{2n} - \begin{bmatrix}G & H\\ K & L\end{bmatrix}\right)\begin{bmatrix}u\\ v\end{bmatrix} =\\ & \begin{bmatrix}\widetilde{e} - Hv\\ e - Ku\end{bmatrix} - \begin{bmatrix}u\\ v\end{bmatrix} + \begin{bmatrix}Gu + Hv\\ Ku + Lv\end{bmatrix} = \begin{bmatrix}\widetilde{e} - (I-G)u\\ e - (I-L)v\end{bmatrix} = V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X), \end{aligned}$$

and substitute it into (9.18) to get

$$V\operatorname{vec}(X^{(k+1)} - X) = \begin{bmatrix}\widehat{u}^{k+1} - u\\ \widehat{v}^{k+1} - v\end{bmatrix}.$$

Finally, using the definition of V in (9.8), we find that

$$\begin{bmatrix} \widehat{u}^{k+1} - u \\ \widehat{v}^{k+1} - v \end{bmatrix} = V \operatorname{vec}(X^{(k+1)} - X) = \begin{bmatrix} X^{(k+1)}\widetilde{q} - X\widetilde{q} \\ X^{(k+1)T}q - X^T q \end{bmatrix} = \begin{bmatrix} X^{(k+1)}\widetilde{q} + \widetilde{e} - u \\ X^{(k+1)T}q + e - v \end{bmatrix}$$

and thus $\widehat{u}^{k+1} = X^{(k+1)}\widetilde{q} + \widetilde{e}$, $\widehat{v}^{k+1} = X^{(k+1)T}q + e$. □

The theorem brings deeper insight into Newton's and Lu's iterations. Lu's iteration can be viewed as a structured Newton's iteration exploiting the displacement structure found in (9.7). Therefore, the two algorithms take the same number of iterations to converge, as the computation they perform is the same. Observe also that the Lu version of this algorithm is slightly more efficient, since it operates on the generators only, and never forms the matrices $X^{(k)}$ explicitly. For this reason, we only present numerical results regarding Lu's iteration.

9.5 Shift technique

As we mentioned before, the case $(c, \alpha) = (1, 0)$ corresponds to a null recurrent NARE. Several drawbacks are encountered when this happens [GH06]. The singularity of the Jacobian does not guarantee the quadratic convergence of Newton's iteration; in fact, Newton's and therefore Lu's method converge linearly. Moreover, as proved in Theorem 6.11 a perturbation $O(\varepsilon)$ in the coefficients of the equation leads to an $O(\sqrt{\varepsilon})$ variation in the solution.

These drawbacks can be removed by means of the shift technique originally introduced by He, Meini and Rhee [HMR02] and applied to Riccati equations in [GIM07] and [BILM06].

The shift technique consists in building a rank-1 modification of the original NARE which has the same minimal solution.

Theorem 9.2 ([GIM07])**.** *Let* (1.5) *be transient or null recurrent, v be the right eigenvector of $\mathcal{H}$ corresponding to the eigenvalue 0, $p \in \mathbb{R}^{m+n}$ be any vector such that $p^T v = 1$, and $\eta > 0$.*

1. *The matrix*
$$\widetilde{\mathcal{H}} = \mathcal{H} + \eta v p^T$$
has the same spectrum of $\mathcal{H}$, except that the eigenvalue 0 is replaced by η.

2. *the minimal solution X of* (1.5) *is also the minimal solution of the NARE with Hamiltonian $\widetilde{\mathcal{H}}$, i.e.,*
$$X\widetilde{C}X - X\widetilde{D} - \widetilde{A}X + \widetilde{B} = 0, \tag{9.19}$$
with
$$\widetilde{A} = A - \eta v_2 q^T, \quad \widetilde{B} = B + \eta v_2 e^T, \quad \widetilde{C} = C - \eta v_1 q^T, \quad \widetilde{D} = D + \eta v_1 e^T. \tag{9.20}$$

Notice that the shifted equation (9.19) is nonsingular when the original equation is transient, and positive recurrent when the original equation is null recurrent; in particular, it is never critical. Its solution can be computed with the usual algorithms faster than the one of the original NARE, as the ratio between the two central eigenvalues is increased.

In this section, we show that it is possible to incorporate this technique in our fast implementation of Newton's method.

Under the assumptions (1.10), (1.11) the right eigenvector of $\mathcal{H}$ corresponding to zero is given by $v = \begin{bmatrix} v_1^T & v_2^T \end{bmatrix}^T$, where $v_1 = \Gamma^{-1}q$, $v_2 = \Delta^{-1}e$. This can be seen by direct inspection using (9.1) and the fact that $e^T\Gamma^{-1}q + q^T\Delta^{-1}e = c = 1$.

We consider the rank-one correction

$$\widetilde{\mathcal{H}} = \mathcal{H} + \eta \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} p^T,$$

where $0 < \eta \leq \gamma_1$ and $p^T = \begin{bmatrix} e^T & q^T \end{bmatrix}$. It holds that $p^T v = 1$, in fact $p^T v = e^T \Gamma^{-1} q + q^T \Delta^{-1} e = 1$.

With the choice $p^T = [e^T \; q^T]$, $\widetilde{\mathcal{H}}$ remains a diagonal plus rank-one matrix, and so is $\widetilde{\mathcal{M}} = \mathcal{K}\widetilde{\mathcal{H}}$; hence, we only need to prove that $\widetilde{\mathcal{M}}$ is an M-matrix to ensure that the algorithms proposed in Sections 9.2 and 9.4 can be applied to (9.19). In fact, we have

$$\widetilde{\mathcal{M}} = \begin{bmatrix} \Gamma & 0 \\ 0 & \Delta \end{bmatrix} - \begin{bmatrix} q - \eta v_1 \\ e + \eta v_2 \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix},$$

and since we chose $0 < \eta \leq \gamma_1 < \gamma_2 < \ldots < \gamma_n$, and $q \geq 0$, then $q - \eta v_1 = (I_n - \eta \Gamma^{-1}) q \geqslant 0$, thus $\widetilde{\mathcal{M}}$ is a Z-matrix. By the Perron–Frobenius theorem applied to $\rho I - \mathcal{M}$, there exists a vector $u > 0$ such that $u^T \mathcal{M} \geq 0$, and $\mu = u^T \mathcal{K} v \geq 0$; therefore,

$$u^T \widetilde{\mathcal{M}} = u^T \mathcal{M} + \eta u^T \mathcal{K} v p^T \geq 0,$$

thus by Lemma 2.3 $\widetilde{\mathcal{M}}$ is an M-matrix.

Newton's iteration applied to equation (9.19) provides a quadratically convergent algorithm of complexity $O(n^2)$ for solving the Riccati equation (1.5) in the critical case. Moreover, since the singularity has been removed, it is expected that X, as minimal solution of (9.19), is better conditioned than with respect to the coefficients of (1.5), and that a higher precision can be reached in the computed solution. This fact is confirmed by the numerical experiments as shown in Section 9.7.

9.6 Numerical stability

Our first concern about numerical stability is proving that the matrix $R^{(k)} = I_{2n} - V \mathcal{D}^{-1} U^{(k)}$ resulting after the application of the SMW formula to the Jacobian $\mathcal{D} - U^{(k)} V$ is well-conditioned whenever the Jacobian is. In the following analysis, we assume that the norm $\left\| (\mathcal{D} - U^{(k)} V)^{-1} \right\|_1$ is bounded, and we drop the superscripts (k) to simplify the notation.

Observe that $0 \leq \mathcal{D}^{-1} \leq (\mathcal{D} - UV)^{-1}$, therefore $\mathcal{D}$ is well-conditioned. Moreover, one has

$$\mathcal{B}^{-1} = \begin{bmatrix} (\mathcal{D} - UV)^{-1} & 0 \\ V(\mathcal{D} - UV)^{-1} & I \end{bmatrix} \begin{bmatrix} I & U \\ 0 & I \end{bmatrix}, \qquad \text{with } \mathcal{B} = \begin{bmatrix} \mathcal{D} & -U \\ -V & I \end{bmatrix};$$

therefore $\mathcal{B}$ is an M-matrix and is well-conditioned. Now, $R = I - V \mathcal{D}^{-1} U$ is the Schur complement of $\mathcal{D}$ in $\mathcal{B}$, and thus R^{-1} is a submatrix of $\mathcal{B}^{-1}$ [GVL96]. This implies $\left\| R^{-1} \right\|_1 \leq \left\| \mathcal{B}^{-1} \right\|_1$, hence R is well-conditioned, too.

Another stability problem could arise from the generator growth during the fast Gaussian elimination step. Generator growth has been reported in some cases with the GKO algorithm [SB95], especially when the starting generators are ill-conditioned. This is not our case, since the starting generators are bounded, and no significant generator growth has been observed during our experiments.

9.7 Numerical experiments

We consider here the same numerical examples as in [GL00] and in [Lu05a]. The sequences t_i and w_i, which appear in the discretization as the nodes and weights of a Gaussian quadrature

	$\alpha = 0.5,\ c = 0.5$		$\alpha = 10^{-8},\ c = 1 - 10^{-6}$		$\alpha = 0,\ c = 1$		
n	Lu	LuF	Lu	LuF	Lu	LuF	LuS
32	2.0×10^{-3}	1.0×10^{-3}	6.0×10^{-3}	2.0×10^{-3}	9.0×10^{-3}	5.0×10^{-3}	3.0×10^{-3}
64	1.5×10^{-3}	6.0×10^{-3}	2.8×10^{-2}	8.0×10^{-3}	4.8×10^{-2}	1.5×10^{-2}	4.0×10^{-3}
128	1.0×10^{-1}	1.5×10^{-2}	2.0×10^{-1}	3.6×10^{-2}	3.6×10^{-1}	6.1×10^{-2}	1.3×10^{-2}
256	7.1×10^{-1}	8.0×10^{-2}	$1.6 \times 10^{+0}$	1.6×10^{-1}	$2.7 \times 10^{+0}$	2.9×10^{-1}	6.4×10^{-2}
512	$8.6 \times 10^{+0}$	5.6×10^{-1}	$1.8 \times 10^{+1}$	$1.3 \times 10^{+0}$	$3.1 \times 10^{+1}$	$2.1 \times 10^{+0}$	3.9×10^{-1}

Table 9.1: Comparison of CPU time in seconds of Lu's algorithm (Lu), its fast version presented here (LuF), and the shifted algorithm in the critical case (LuS)

	$\alpha = 0.5,\ c = 0.5$		$\alpha = 0,\ c = 1$		
n	Lu	LuF	Lu	LuF	LuS
32	4.8×10^{-16} (4)	2.3×10^{-16} (5)	5.2×10^{-8} (25)	4.2×10^{-8} (26)	4.4×10^{-16} (6)
256	1.6×10^{-15} (4)	4.0×10^{-16} (5)	4.6×10^{-8} (25)	8.0×10^{-8} (25)	1.2×10^{-15} (6)

Table 9.2: Comparison of the relative error (and in parentheses the number of steps) of Lu's algorithm (Lu), its fast version presented here (LuF), and the shifted algorithm in the critical case (LuS)

method, are obtained by dividing the interval $[0, 1]$ into $\frac{n}{4}$ subintervals of equal length, and by applying to each one the 4-nodes Gauss-Legendre quadrature.

The computation was performed with three different choices of the parameters (c, α), namely, $(0.5, 0.5)$, $(1 - 10^{-6}, 10^{-8})$, and $(1, 0)$. The first example is far from the singular case and poses little numerical trouble; the second one is numerically close to the critical case, but still nonsingular; the third example is the critical case of (1.10).

We have compared the method described here with the original algorithm presented in [Lu05a], which does not exploit the Trummer-like structure and requires a traditional $O(n^3)$ algorithm to solve (9.15). For this purpose we used the LAPACK function `dgesv`. In the critical case, we also considered the algorithm described here coupled with the shift technique of Section 9.5.

The three algorithms were implemented in Fortran 90 and the tests were performed using the Lahey Fortran compiler on a 2.8 GHz Xeon biprocessor.

In Table 9.1 we compare the timing of the original unstructured Lu's algorithm with its fast $O(n^2)$ version. The numerical results highlight the different order of complexity. Observe that in the critical case the shift technique reduces the timings even further.

In Table 9.2 we compare the relative error of the two methods and the number of steps required. Here the error is computed as $\left\|\widetilde{X} - X\right\|_1 / \|X\|_1$, where $\widetilde{X}$ and X are the solution computed in double and in quadruple precision, respectively.

The stopping criterion is based on the computation of

$$\text{Res} := \frac{\|u_k - u_{k-1}\|_1 + \|v_k - v_{k-1}\|_1}{2}.$$

Observe that the cost of computing Res is negligible.

As one can see, in the critical case the accuracy of the solution obtained with the non-shifted algorithms is of the order of $O(\sqrt{\varepsilon})$, where ε is the machine precision, in strict accordance with Theorem 6.11. The speed-up obtained is greater than 2 even for small values of n. In the critical case with size $n = 512$ our algorithm is about 80 times faster than Lu's original algorithm. The problems deriving from the large number of steps and the poor accuracy are completely removed by the shift technique.

9.8 Conclusions and research lines

The implementation of the Lu-Newton iteration provided here reduces the complexity of the Newton algorithm from $O(n^3)$ per step to $O(n^2)$ per step, with great impact on the computational times for the solution of this equation. The method presented in this chapter, which was published in [BIP08] outperforms all algorithms present in literature.

Now that the Trummer-like and Cauchy-like structure of the iterates has been exposed, a challenging issue is to prove that Lu's iteration for this problem can be effectively computed with less than $O(n^2)$ ops. There exist algorithms for computing the Cauchy matrix–vector product [Ger88] and for approximating the inverse of a Cauchy matrix [MRT05] with $O(n \log^2 n)$ ops (*superfast algorithms*), even though they not exempt from numerical issues. If the implementation can be carried on with sufficient efficiency and stability, this could lead to another great speedup in the solution of this class of equations.

Another natural question is whether SDA and the cyclic reduction algorithm presented in Chapter 7 can be applied in a structure-preserving implementation. In another paper, we proved that both preserve the Cauchy-like structure of the iterates, and provided a $O(n^2)$ implementation [BMP09]. However, the numerics for these algorithms are not satisfying: the computational cost per step is higher than the one for the Lu/Newton method, and serious stability problems arise for close-to-critical equations. Despite several attempts, which led as a byproduct to the results in Chapter 8, we did not manage to remove such instability.

Thus, surprisingly, while SDA and CR are more effective than Newton methods for general algebraic Riccati equations, the situation is reversed for this rank-structured problem.

The algorithms exposed here can be extended to the case where $\mathcal{M}$ is diagonal plus rank r; in this case, the computational cost grows as $O(r^3 n^2)$. On the other hand, the corresponding extensions of the structured SDA and CR [BMP09] have computational cost $O(r^2 n^2)$.

Part III

Matrix equations from control theory

Chapter 10

Lur'e equations

10.1 Introduction

For given matrices $A, Q \in \mathbb{C}^{n\times n}$ with $Q = Q^*$ and $B, C \in \mathbb{C}^{n\times m}$, $R \in \mathbb{C}^{m\times m}$, we consider the *Lur'e equations*

$$\begin{aligned} A^*X + XA + Q &= K^*K, \\ XB + C &= K^*L, \\ R &= L^*L, \end{aligned} \tag{10.1}$$

that have to be solved for $(X, K, L) \in \mathbb{C}^{n\times n} \times \mathbb{C}^{p\times n} \times \mathbb{C}^{p\times m}$ with $X = X^*$ and p as small as possible. Equations of type (10.1) were first introduced by A.I. Lur'e [Lur51] in 1951 (see [Bar07] for an historical overview) and play a fundamental role in systems theory, since properties like dissipativity of linear systems can be characterized via their solvability [And66, AN68, AV73, Wil72]. This type of equations moreover appears in the infinite time horizon linear-quadratic optimal control problem [CAM77, CA77, CA78, Yak85, ZDG96], spectral factorization [CG89, CALM97] as well as in balancing-related model reduction [CW95, GA04, OJ88, PDS02, RS10]. In the case where R is invertible, the matrices K and L can be eliminated by obtaining the algebraic Riccati equation

$$A^*X + XA - (XB + C)R^{-1}(XB + C)^* + Q = 0. \tag{10.2}$$

Whereas this type is well explored both from an analytical and numerical point of view [LR95, Wil71, Ben97], the case of singular R has received comparatively little attention. However, the singularity of R is often a structural property of the system to be analyzed [RS08] and can therefore not be avoided by arguments of genericity.

Two approaches exist for the numerical solution of Lur'e equations with (possibly) singular R. The works [SJ71, WWS94] present an iterative technique for the elimination of variables corresponding to $\ker R$. After a finite number of steps this leads to a Riccati equation. This also gives an equivalent solvability criterion that is obtained by the feasibility of this iteration. The most common approach to the solution of Lur'e equations is the slight perturbation of R by εI_m for some $\varepsilon > 0$. Then by using the invertibility of $R + \varepsilon I$, the corresponding perturbed Lur'e equations are now equivalent to the Riccati equation

$$A^*X_\varepsilon + X_\varepsilon A - (XB + C)(R + \varepsilon I)^{-1}(X_\varepsilon B + C)^* + Q = 0. \tag{10.3}$$

It is shown in [JS71, Tre87] that certain corresponding solutions X_ε converge to a solution of (10.1). Whereas the first approach has the great disadvantage that it relies on successive null

space computations (which may be arbitrarily an ill-conditioned numerical problem), the big problem of the perturbation approach is that there exist no estimates for the perturbation error $|X - X_\varepsilon|$ and, furthermore, the numerical condition of the Riccati equation (10.3) increases drastically as ε tends to 0. From a theoretical point of view, Lur'e equations have been investigated in [CALM97, Bar07]. The solution set is completely characterized in [Rei09] via the consideration of the matrix pencil (1.13). This pencil has the special property of being *even*, that is $\mathcal{E}$ is skew-Hermitian and $\mathcal{A}$ is Hermitian. Solvability of (10.1) is characterized via the eigenstructure of this pencil. It is furthermore shown that there exists some correspondence to *deflating subspaces* of (1.13). Under some slight additional conditions of the pair (A, B), it is shown in [Rei09] that there exists a so-called *maximal solution* X. In this case, maximal means that X is, in terms of semi-definiteness, above all other solutions of the Lur'e equations. This solution is of particular interest in optimal control as well as in model reduction.

In this chapter, we set up an iterative method for Lur'e equations that converges linearly to the maximal solution, based on SDA and the theoretical results of [Rei09].

10.2 Solvability of Lur'e equations

In this part we collect theoretical results that characterize the solvability of Lur'e equations. For convenience, we call a Hermitian matrix X a solution of the Lur'e equations if (10.1) is fulfilled for some $K \in \mathbb{C}^{p\times n}$, $L \in \mathbb{C}^{p\times m}$.

We now introduce some further concepts which are used to characterize solvability of the Lur'e equations.

For the Lur'e equations (10.1), the *spectral density function* is defined as

$$\Phi(i\omega) := \begin{bmatrix}(i\omega I - A)^{-1}B \\ I_m\end{bmatrix}^* \begin{bmatrix}Q & C \\ C^* & R\end{bmatrix} \begin{bmatrix}(i\omega I - A)^{-1}B \\ I_m\end{bmatrix} \tag{10.4}$$

In several works, the spectral density function is also known as *Popov function*. The *linear matrix inequality (LMI)* associated with the Lur'e equations (10.1) is defined as

$$\begin{bmatrix}A^*Y + YA + Q & YB + C \\ B^*Y + C^* & R\end{bmatrix} \geq 0. \tag{10.5}$$

We recall that in this chapter we use the symbol $\geq$ to denote the positive definite ordering. The solution set of the LMI is defined as

$$\mathcal{S}_{LMI} := \{Y \in \mathbb{C}^{n\times n} : Y \text{ is Hermitian and (10.5) holds true}\}. \tag{10.6}$$

The LMI (10.5) is called *feasible* if $\mathcal{S}_{LMI} \neq \emptyset$. It can be readily verified that $Y \in \mathcal{S}_{LMI}$ solves the Lur'e equations if it minimizes the rank of (10.5).

We now collect some known equivalent solvability criteria for Lur'e equations. In the following we require that the pair (A, B) is stabilizable. Note that this assumption can be replaced by the weaker condition of *sign-controllability* [Rei09].

Theorem 10.1 ([Rei09])**.** *Let the Lur'e equations (10.1) with associated even matrix pencil $s\mathcal{E} - \mathcal{A}$ as in (1.13) and spectral density function Φ as in (10.4) be given. Assume that at least one of the claims*

(i) the pair (A, B) is stabilizable and the pencil $s\mathcal{E} - \mathcal{A}$ as in (1.13) is regular;

(ii) the pair (A, B) is controllable;

holds true. Then the following statements are equivalent:

1. *There exists a solution (X, K, L) of the Lur'e equations.*
2. *The LMI (10.5) is feasible*
3. *For all $\omega \in \mathbb{R}$ with $i\omega \notin \sigma(A)$ holds $\Phi(i\omega) \geq 0$;*
4. *In the EKCF of $s\mathcal{E} - \mathcal{A}$, all blocks of type E2 have positive signature and even size, and all blocks of type E3 have negative signature and odd size.*
5. *In the EKCF of $s\mathcal{E} - \mathcal{A}$, all blocks of type E2 have even size, and all blocks of type E3 have negative signature and odd size.*

In particular, solutions of the Lur'e equations fulfill $(X, K, L) \in \mathbb{C}^{n\times n} \times \mathbb{C}^{n\times p} \times \mathbb{C}^{m\times p}$ with $p = \operatorname{normalrank} \Phi$.

It is shown in [Rei09] that $m - \operatorname{normalrank} \Phi$ is the number of blocks of type E4 in an EKCF of $s\mathcal{E} - \mathcal{A}$. In particular, the pencil $s\mathcal{E} - \mathcal{A}$ is regular if and only if Φ has full normal rank.

We can identify a *maximal solution* among the solutions of the Lur'e equations.

Theorem 10.2 ()**.** *Let the Lur'e equations (10.1) be given with stabilizable pair (A, B). Assume that $\mathcal{S}_{LMI}$ as defined in (10.6) is non-empty. Then there exists a solution X_+ of the Lur'e equations that is maximal in the sense that for all $Y \in \mathcal{S}_{LMI}$ it holds $Y \leq X_+$.*

Let $\mathcal{M} \in \mathbb{C}^{k\times k}$ be given. A subspace $\mathcal{V} \subset \mathbb{C}^k$ is called *$\mathcal{M}$-neutral* if $x^*\mathcal{M}y = 0$ for all $x, y \in \mathcal{V}$. Note that the notion of $\mathcal{E}$-neutrality, which is used in the following result, is the counterpart of Lagrangianness in this setting.

Theorem 10.3 ([Rei09])**.** *Let the Lur'e equations (10.1) be given with stabilizable pair (A, B). Assume that $\mathcal{S}_{LMI}$ as defined in (10.6) is non-empty. Then*

$$\begin{bmatrix} X_+ & 0 \\ I_n & 0 \\ 0 & I_m \end{bmatrix} \tag{10.7}$$

spans the unique $(n+m)$-dimensional semi-c-stable $\mathcal{E}-$neutral subspace of the pencil (1.13)*.*

The above theorem states that the maximal solution can be expressed in terms of a special deflating subspace of $s\mathcal{E} - \mathcal{A}$. This space can be constructed from the matrix $U \in \mathbb{C}^{2n+m\times 2n+m}$ bringing the pencil $s\mathcal{E} - \mathcal{A}$ into even Kronecker form (2.4). Considering the partitioning $U = [\, U_1, \ldots, U_k \,]$ according to the block structure of the EKCF, a matrix $V \in \mathbb{C}^{2n+m\times n+m}$ spanning the desired deflating subspace is given by

$$V = \begin{bmatrix} V_1 & \ldots & V_k \end{bmatrix} \quad \text{for } V_j = U_j Z_j, \tag{10.8}$$

where

$$Z_j := \begin{cases} [\, I_{k_j} \,,\, 0_{k_j} \,]^T, & \text{if } \mathcal{D}_j \text{ is of type E1,} \\ [\, I_{k_j/2} \,,\, 0_{k_j/2} \,]^T, & \text{if } \mathcal{D}_j \text{ is of type E2,} \\ [\, I_{(k_j+1)/2} \,,\, 0_{(k_j-1)/2} \,]^T, & \text{if } \mathcal{D}_j \text{ is of type E3,} \\ [\, I_{k_j} \,,\, 0_{k_j+1} \,]^T, & \text{if } \mathcal{D}_j \text{ is of type E4.} \end{cases}$$

In particular, the desired subspace contains all the vectors belonging to the Kronecker chains relative to c-stable eigenvalues, no vectors from the Kronecker chains relative to c-unstable eigenvalues, the first $k_j/2$ vectors from the chains relative to c-critical eigenvalues, and the first $(k_j + 1)/2$ from the chains relative to eigenvalues at infinity.

10.3 A reduced Lur'e pencil

Our aim in this chapter is applying SDA (Algorithm 8) to the solution of the Lur'e equations (10.1). If R were nonsingular, the most linear approach would be to transform the Lur'e equations to a (continuous-time) Riccati equation, and then using the usual strategy of performing a Cayley transform followed by a SSF-I factorization in order to get the proper d-splitting and the SSF-I pencil which is needed in the SDA initialization. However, as we saw in the introduction, when R is singular it is not possible to reduce the Lur'e equations (10.1) to an algebraic Riccati equation. In this section, we show that if we change the order of the steps and start with the Cayley transform, then this problem disappears.

Here and in the following, we suppose that the pencil (1.13) is regular.

Let $s\mathcal{E} - \mathcal{A}$ be the pencil in (1.13). By Theorem 6.19, the SSF-I form of its Cayley transform (with dimensions chosen such that $E \in \mathbb{C}^{n\times n}$, $F \in \mathbb{C}^{(m+n)\times(m+n)}$) is given by

$$\begin{bmatrix} E & -G \\ -H & F \end{bmatrix} = \begin{bmatrix} 0 & A-\gamma I & B \\ A^*-\gamma I & Q & C \\ B^* & C^* & R \end{bmatrix}^{-1} \begin{bmatrix} 0 & A+\gamma I & B \\ A^*+\gamma I & Q & C \\ B^* & C^* & R \end{bmatrix}. \tag{10.9}$$

Let now

$$\widetilde{A} := \begin{bmatrix} 0 & A-\gamma I \\ A^*-\gamma I & Q \end{bmatrix},$$

and assume that both $\widetilde{A}$ and its Schur complement

$$R - \begin{bmatrix} B^* & C^* \end{bmatrix} \widetilde{A}^{-1} \begin{bmatrix} B \\ C \end{bmatrix} = \Phi(\gamma)$$

(here Φ is the same function as in (10.4), as one can verify by expanding both expressions) are nonsingular. In this case we can perform the inversion with the help of a block LDU factorization. Tedious computations lead to a matrix of the form

$$\begin{bmatrix} \widehat{A} & 0 \\ \widehat{B} & I_m \end{bmatrix},$$

with

$$\widehat{A} = I + 2\gamma\widetilde{A}^{-1}S + 2\gamma\widetilde{A}^{-1} \begin{bmatrix} B \\ C \end{bmatrix} \Phi(\gamma)^{-1} \begin{bmatrix} B^* & C^* \end{bmatrix} \widetilde{A}^{-1}S, \qquad S = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}. \tag{10.10}$$

In the blocks used in SSF-I, this means that

$$F = \begin{bmatrix} \widehat{F} & 0 \\ * & I_m \end{bmatrix}, \quad G = \begin{bmatrix} \widehat{G} & 0 \end{bmatrix}, \quad H = \begin{bmatrix} \widehat{H} \\ * \end{bmatrix},$$

where the blocks $\widehat{F}$, $\widehat{G}$, $\widehat{H}$ have size $n \times n$. It follows that a special right deflating subspace of this pencil is

$$\begin{bmatrix} 0_{2n\times m} \\ I_m \end{bmatrix},$$

whose only eigenvalue is 1 with algebraic and geometric multiplicity m, while the deflating subspaces relative to the rest of the spectrum are in the form

$$\begin{bmatrix} V \\ * \end{bmatrix},$$

where V has $2n$ rows and is a deflating subspace of the reduced pencil

$$s\begin{bmatrix} I_n & -\widehat{G} \\ 0 & \widehat{F} \end{bmatrix} - \begin{bmatrix} E & 0 \\ -\widehat{H} & I_n \end{bmatrix}. \tag{10.11}$$

Using (10.10) and the fact that $\widetilde{A}$ and $\Phi(\gamma)$ are Hermitian, one sees that $\widehat{A}S$ is Hermitian, too. This means that $E^* = F$ and $G = G^*$, $H = H^*$, that is, the pencil (10.11) is symplectic.

The pencil (10.11) is given by $P^*(s\mathcal{E} - \mathcal{A})P$, where P is the projection on

$$\left(\operatorname{Span}\left(\begin{bmatrix} 0 \\ 0 \\ I_m \end{bmatrix}\right)\right)^{\perp} = (\ker \mathcal{E})^{\perp}.$$

With this characterization, it is easy to derive the KCF of (10.11) from that of the Cayley transform of (1.13). We see that $\ker \mathcal{E}$ is the space spanned by the first column of each W2 block (as a corollary, we see that there are exactly $m = \dim \ker E$ such blocks). These blocks are transformed into blocks W1 with $\lambda = 1$ by the Cayley transform. Thus projecting on their orthogonal complement corresponds to dropping the first row and column from each of the W1 blocks relative to $\lambda = 1$. In particular, it follows that if the criteria in Theorem 10.1 hold, then every Kronecker block of (10.11) relative to an eigenvalue λ on the unit circle has even size. Therefore, the reduced pencil (10.11) is weakly d-split. By considering which vectors are needed from each Kronecker chain to form the subspace in (10.8), we get therefore the following result.

Theorem 10.4 ([Rei09])**.** *Let $\mathcal{V}$ be the unique $(n+m)$-dimensional c-semi-stable $\mathcal{E}$-neutral deflating subspace of* (1.13)*. Then, there is a matrix $V_2 \in \mathbb{C}^{n\times m}$ such that*

$$\mathcal{V} = \operatorname{Span}\begin{bmatrix} V_1 & 0 \\ V_2 & I_m \end{bmatrix},$$

where V_1 spans the canonical d-semi-unstable subspace of the pencil (10.11)*. Moreover, if* $\operatorname{Span}(V_1)$ *admits a basis in the form*

$$\begin{bmatrix} X_+ \\ I_n \end{bmatrix},$$

then X_+ is the maximal solution of the Lur'e equation (10.1)*.*

In other words, X_+ is the canonical weakly stabilizing solution of the DARE

$$X = EX(I - \widehat{H}X)^{-1}E^* + \widehat{G}. \tag{10.12}$$

10.4 Implementation of SDA

Based on the results of the previous sections, we can use SDA to compute the solution to a Lur'e equation. The resulting algorithm is reported as Algorithm 16. The explicit computation of (a possible choice of) K and L is typically not needed in the applications. If the two matrices are needed, they can be computed using the fact that

$$\begin{bmatrix} A^*X + X^*A + Q & XB + C \\ B^*X^* + C^* & R \end{bmatrix} = \begin{bmatrix} K^* \\ L^* \end{bmatrix}\begin{bmatrix} K & L \end{bmatrix} \tag{10.13}$$

is a full rank decomposition.

Algorithm 16: A SDA for the maximal solution of a Lur'e equation

input: A, B, C, Q, R defining a Lur'e equation
output: The weakly stabilizing solution X (and optionally K and L)
Choose a suitable $\gamma > 0$
Compute

$$T \longleftarrow \begin{bmatrix} 0 & A-\gamma I & B \\ A^*-\gamma I & Q & C \\ B^* & C^* & R \end{bmatrix}^{-1} \begin{bmatrix} 0 & A+\gamma I \\ A^*+\gamma I & Q \\ B^* & C^* \end{bmatrix}$$

Partition

$$T = \begin{bmatrix} E & -G \\ -H & E^* \\ * & * \end{bmatrix}$$

Use `SDA` on E, $F = E^*$, G,H to compute G_∞, H_∞
return $X \leftarrow G_\infty$
if K and L are needed **then**
Compute Σ and V^* corresponding to the first m singular vectors of the SVD of (10.13)
return $\begin{bmatrix} K & L \end{bmatrix} \longleftarrow \Sigma^{1/2} V^*$
end if

SDA and symplectic pencils

While we have exposed SDA in its application to nonsymmetric Riccati equations, the algorithm was originally born for discrete- and continuous-time Riccati equation in control theory [HCL05, CFL05]. One of its main advantages there is that it is able to preserve the symplectic structure of the SSF-I pencil.

As the matrix $\mathcal{H}$ associated with a continuous-time algebraic Riccati equation is a Hamiltonian matrix, its Cayley transform is symplectic. Therefore SDA for control theory problems need only work with symplectic matrices and pencils [CFL05]. An easy check shows that a pencil in SSF-I is symplectic if and only if $E = F^T$, $G = G^T$ and $H = H^T$. If this property holds, then all the iterates generated by SDA are symplectic as well, i.e., it holds

$$E_k = F_k^T, \qquad G_k = G_k^T, \qquad H_k = H_k^T$$

for all $k \geq 0$. In particular, at each steps the two matrices F_{k+1} and E_{k+1} are one the transposed of the other; therefore, only the computation of the first is needed. Similarly, the computation and inversion of $I - H_k G_k$ can be avoided, since this matrix is the transposed of $I - G_k H_k$ which is already computed and inverted during the algorithm. If the inversion is performed implicitly (e.g., with a LU factorization), the picture does not change.

In particular, this implies that the symplectic eigensymmetry is preserved during the iteration. This property is crucial, since the eigenvalue condition number for structured perturbations can be substantially lower than the one for unstructured perturbation [KKT06]. Therefore, in presence of Hamiltonian or symplectic eigensymmetry, structure-preserving algorithms yield usually much better results than unstructured ones [BKM05].

Choice of the parameter in the Cayley transform

The accuracy of the computed solution depends also on an appropriate choice of γ. Clearly, two goals have to be considered:

1. the matrix to invert in (10.9) should be well-conditioned;

2. the condition number of the resulting problem, i.e, the separation between the stable and unstable subspace of the Cayley-transformed pencil (10.11), should not be too small.

While the impact of the first factor is easy to measure, the second one poses a more significant problem, since there are no *a priori* estimates for the conditioning of a subspace separation problem. Nevertheless, we may try to understand how the choice of the parameter γ of the Cayley transform affects the conditioning. Roughly speaking, the conditioning of the invariant subspace depends on the distance between its eigenvalues and those of the complementary subspace [GVL96]. The eigenvalues of the transformed pencil are given by $\frac{\lambda-\gamma}{\lambda+\gamma} = 1 - \frac{2\gamma}{\lambda+\gamma}$, thus they tend to cluster around 1 for small values of γ, which is undesirable. The closest to 1 is the one for which $\lambda + \gamma$ has the largest modulus; we may take as a crude approximation $\rho(A) + \gamma$, which can be further approximated loosely with $\|A\|_1 + \gamma$. Therefore, as a heuristic to choose a reasonable value of γ, we may look for a small value of

$$f(\gamma) = \max\left(\text{condest}\left(\begin{bmatrix}\mathcal{E}_1 & \mathcal{A}_2\end{bmatrix}\right), \frac{\|A\|_1 + \gamma}{2\gamma}\right),$$

where condest$(\cdot)$ is the condition number estimate given by Matlab. Following the strategy of [CFL05], we chose to make five steps of the golden section search method [LY08] on $f(\gamma)$ in order to get a reasonably good value of the objective function without devoting too much time to this ancillary computation.

However, we point out that in [CFL05], a different $f(\gamma)$ is used, which apparently only takes into account the first of our two goals. The choice of the function is based on an error analysis of their formulas for the starting blocks of SDA. Due to Theorem 6.25, this part of the error analysis can be simplified to checking condest $\left(\begin{bmatrix}\mathcal{E}_1 & \mathcal{A}_2\end{bmatrix}\right)$.

10.5 Numerical experiments

We have implemented Algorithm 16 (SDA-L) using Matlab, and tested it on the following test problems.

P1 a Lur'e equation with a random stable matrix $A \in \mathbb{R}^{n\times n}$, a random $C = B$, $Q = 0$ and R the $m \times m$ matrix with all the entries equal to 1, with $\text{rk}(R) = 1$. Namely, B was generated with the Matlab command `B=rand(n,m)`, while to ensure a stable A a we used the following more complex sequence of commands: `V=randn(n); W=randn(n); A=-V*V'-W+W';`

P2 a set of problems motivated from real-world examples, taken with some modifications from the benchmark set `carex` [BLM95a]. Namely, we took Examples 3 to 6, which are a set of real-world problems arising from various applications, varying in size and numerical characteristics, and changed the value of R to get a singular problem. In the original versions of all examples, R is the identity matrix of appropriate size; we simply replaced its $(1, 1)$ entry with 0, in order to get a singular problem.

Table 10.1: Relative residual for P1

n	m	SDA-L	R+S			R+N
			$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-8}$	$\varepsilon = 10^{-12}$	$\varepsilon = 10^{-8}$
10	3	1×10^{-15}	2×10^{-8}	4×10^{-10}	3×10^{-6}	3×10^{-10}
50	5	3×10^{-14}	8×10^{-9}	1×10^{-7}	2×10^{-1}	4×10^{-10}
500	10	7×10^{-14}	2×10^{-9}	1×10^{-7}	1×10^{-1}	$\star$

P3 a highly ill-conditioned high-index problem with $m = 1$, $A = I_n + N_n$, $B = e_n$ (the last vector of the canonical basis for $\mathbb{R}^n$), $C = -B$, $R = 0$, $Q = -\operatorname{tridiag}(1, 2, 1)$. Such a problem corresponds to a Kronecker chain of length $2n + 1$ associated with an infinite eigenvalue, and its canonical semi-stable solution is $X = I$. Notice that the conditioning of the invariant subspace problem in this case is $\epsilon^{1/(2n+1)}$, for an unstructured perturbation of the input data of the order of the machine precision ϵ [GLR06, Section 16.5].

The results of SDA-L are compared to those of a regularization method as the one described in (10.3), for different values of the regularization parameter ε. After the regularization, the equations are solved using Algorithm 8 after a Cayley transform with the same parameter γ (R+S), or with the matrix sign method with determinant scaling [Meh91, Hig08] (R+N). We point out that the control toolbox of Matlab contains a command `gcare` that solves a so-called generalized continuous-time algebraic Riccati equation based on a pencil in a form apparently equivalent to that in (1.13). In fact, this command is not designed to deal with a singular R, nor with eigenvalues numerically on the imaginary axis. Therefore, when applied to nearly all the following experiments, this command fails reporting the presence of eigenvalues too close to the imaginary axis.

For the problem P3, where an analytical solution $X = I$ is known, we reported the values of the forward error

$$\frac{\left\|\tilde{X} - X\right\|_F}{\|X\|_F}.$$

For P1 and P2, for which no analytical solution is available, we computed instead the relative residual of the Lur'e equations in matrix form

$$\frac{\left\|\begin{bmatrix} A'X + XA + Q & XB + C \\ B^*X^* + C^* & R \end{bmatrix} - \begin{bmatrix} K^* \\ L^* \end{bmatrix} \begin{bmatrix} K & L \end{bmatrix}\right\|_F}{\left\|\begin{bmatrix} A'X + XA + Q & XB + C \\ B^*X^* + C^* & R \end{bmatrix}\right\|_F}$$

(see (10.13)). The results are reported in Tables 10.1, 10.2 and 10.3. A star $\star$ in the data denotes convergence failure.

We see that in all the experiments our solution method obtains a better result than the ones based on regularization.

Table 10.2: Relative residual for P2

Problem number	SDA-L	R+S			R+N
		$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-8}$	$\varepsilon = 10^{-12}$	$\varepsilon = 10^{-8}$
3	6×10^{-16}	1×10^{-7}	9×10^{-10}	8×10^{-6}	1×10^{-9}
4	9×10^{-16}	6×10^{-7}	6×10^{-9}	2×10^{-7}	6×10^{-9}
5	6×10^{-15}	3×10^{-7}	1×10^{-9}	3×10^{-8}	1×10^{-9}
6	2×10^{-15}	6×10^{-12}	2×10^{-12}	1×10^{-11}	4×10^{-13}

Table 10.3: Forward error for P3

n	SDA-L	R+S			R+N
		$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-8}$	$\varepsilon = 10^{-12}$	$\varepsilon = 10^{-8}$
1	1×10^{-8}	1×10^{-3}	1×10^{-4}	1×10^{-6}	1×10^{-4}
2	5×10^{-5}	3×10^{-2}	1×10^{-2}	3×10^{-2}	$\star$
3	2×10^{-3}	1×10^{-1}	5×10^{-2}	$2 \times 10^{+0}$	$\star$
4	1×10^{-2}	4×10^{-1}	1×10^{-1}	8×10^{-1}	$\star$
5	6×10^{-2}	$1 \times 10^{+0}$	4×10^{-1}	$2 \times 10^{+0}$	$\star$

10.6 Conclusions and research lines

In this chapter we have presented a new numerical method for the solution of Lur'e matrix equations. Unlike previous methods based on regularization, this approach allows one to solve the original equation without introducing any artificial perturbation. The method is exposed in a work in preparation [PR].

The first step of this approach is applying a Cayley transform to convert the problem to an equivalent discrete-time pencil. In this new form, the infinite eigenvalues can be easily deflated, reducing the problem to a discrete-time algebraic Riccati equation with eigenvalues on the unit circle. For the solution of this latter equation, the structured-preserving doubling algorithm was chosen, due to its good behavior in presence of eigenvalues on the unit circle, as proved by the convergence results in [HL09]. Direct methods, such as the symplectic eigensolvers presented in [Fas00], could also be used for the solution of the deflated DARE.

The numerical experiments confirm the effectiveness of our new approach for regular matrix pencils. It is not clear whether the same method can be adapted to work in cases in which the pencil (1.13) is not regular, which may indeed happen in the context of Lur'e equations. Another issue is finding a method to exploit the low-rank structure of Q (when present). These further developments are currently under our investigation.

Chapter 11

Generalized SDA

11.1 Introduction

The traditional hypotheses needed for the implementation of SDA are that the two Riccati equations

$$Q + A^T X + XA - XGX = 0$$
$$YQY + YA^T + AY - G = 0$$

have respectively a stabilizing and anti-stabilizing solution. This is equivalent to saying that the Hamiltonian

$$\mathcal{H} = \begin{bmatrix} A & -G \\ -Q & -A^T \end{bmatrix}$$

has stable and unstable invariant subspaces in the form

$$U = \begin{bmatrix} U^{(1)} \\ U^{(2)} \end{bmatrix}, \quad V = \begin{bmatrix} V^{(1)} \\ V^{(2)} \end{bmatrix},$$

with $U^{(1)}$ and $V^{(2)}$ nonsingular. If this holds, we can form the desired solutions to the two Riccati equations as $X = U_2 U_1^{-1}$, $Y = V_1 V_2^{-1}$.

The case in which $\mathcal{H}$ has eigenvalues on the imaginary axis arises as well in the applications [FD87, CG89]; in this case, one looks for the unique semi-stable (semi-unstable) Lagrangian subspaces U (V). SDA converges in this case as well, as has recently been proved in [HL09], but its convergence turns to linear instead of quadratic.

In the case in which one or both of the blocks $U^{(1)}$ and $V^{(2)}$ is ill-conditioned, the corresponding Riccati solution X or Y becomes very large; the quantities appearing in the algorithm grow as well, and ill-conditioning in the intermediate steps often leads to poor performance in the algorithm. In fact, the algorithm uses the initial data of the control problem only in its initialization; thus, inaccuracy in one of the intermediate steps leads to a loss of accuracy that is impossible to counter in later steps.

In this chapter, we address this situation, and propose a generalization of SDA that aims to attain more accuracy in these problematic cases, at the expense of a larger cost per step. Numerical examples are provided that show how our SDA variant achieves better results on several border-case problems.

11.2 Generalized SDA

Our aim is building a variant of SDA in which we drop the condition that the bases for the canonical subspaces are in the form (6.23), in the hope to get an algorithm which does not break down when such form cannot be constructed. To do so, we relax the assumption on SSF-I, and we aim for a pencil of the form

$$\mathcal{A} = \begin{bmatrix} E & 0 \\ H^{(1)} & H^{(2)} \end{bmatrix}, \quad \mathcal{E} = \begin{bmatrix} G^{(1)} & G^{(2)} \\ 0 & F \end{bmatrix}, \tag{11.1}$$

which we call weak SSF-I. Unlike SSF-I, there is not an unique weak SSF-I pencil which is right-similar to a given regular pencil. In fact, we can multiply on the left both $\mathcal{A}$ and $\mathcal{E}$ by a matrix in the form

$$\begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix}$$

without altering the weak SSF-I. Thus, a normalization choice is needed along the algorithm that we are constructing. The choice of such normalization and its consequences are discussed in the following sections.

Derivation of the general form

We replay the steps that lead to SDA starting with the weak SSF-I. If we impose

$$\begin{bmatrix} \widetilde{E}_{11} & \widetilde{A}_{12} \\ \widetilde{E}_{21} & \widetilde{A}_{22} \end{bmatrix} = I_{n+m}, \tag{11.2}$$

Theorem 6.14 yields

$$\begin{bmatrix} \widetilde{E} & \widetilde{G} \\ \widetilde{H} & \widetilde{F} \end{bmatrix} = \begin{bmatrix} E & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} G^{(1)} & G^{(2)} \\ H^{(1)} & H^{(2)} \end{bmatrix}^{-1}.$$

Thus, setting

$$\begin{bmatrix} G^{(1)} & G^{(2)} \\ H^{(1)} & H^{(2)} \end{bmatrix}^{-1} := \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}, \tag{11.3}$$

we obtain an analogous of (6.22)

$$\widehat{\mathcal{A}} = \begin{bmatrix} EZ_{11}E & 0 \\ H^{(1)} - FZ_{12}E & H^{(2)} \end{bmatrix}, \quad \widehat{\mathcal{E}} = \begin{bmatrix} G^{(1)} & G^{(2)} - EZ_{21}F \\ 0 & FZ_{22}F \end{bmatrix}. \tag{11.4}$$

Remark 11.1. Notice that this update relation plays well with the normalization ambiguity in the weak SSF-I form: in fact, if we start over from

$$\begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix} \mathcal{A} = \begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix} \begin{bmatrix} E & 0 \\ H^{(1)} & H^{(2)} \end{bmatrix},$$
$$\begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix} \mathcal{E} = \begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix} \begin{bmatrix} G^{(1)} & G^{(2)} \\ 0 & F \end{bmatrix}$$

and follow the same steps, we get to

$$\begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix} \widehat{\mathcal{A}}, \qquad \begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix} \widehat{\mathcal{E}}. \tag{11.5}$$

Similarly, if we replace (11.2) with the more general form

$$\begin{bmatrix} \widetilde{E}_{11} & \widetilde{A}_{12} \\ \widetilde{E}_{21} & \widetilde{A}_{22} \end{bmatrix} = \begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix} \tag{11.6}$$

the multiplicative factors introduced can be factored out as in (11.5).

Alternatively, we may compute the CS-AB solution as in (11.4), without any normalizing factor, and only later multiply it by a factor as in (11.5) to impose a suitable constraint.

By iterating the pencil transformation described here, we get Algorithm 17. For the sake of brevity, in the algorithm and the following we set

$$\mathcal{H}_k := \begin{bmatrix} H_k^{(1)} & H_k^{(2)} \end{bmatrix}, \qquad \mathcal{G}_k := \begin{bmatrix} G_k^{(1)} & G_k^{(2)} \end{bmatrix}.$$

Algorithm 17: gSDA

input: E_0, F_0, G_0, H_0 defining a pencil in SSF-I
output: U,V spanning respectively the canonical semi-d-stable and semi-d-unstable subspace

$k \longleftarrow 0$
while a suitable stopping criterion is not satisfied **do**
$\begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \longleftarrow \begin{bmatrix} G_k^{(1)} & G_k^{(2)} \\ H_k^{(1)} & H_k^{(2)} \end{bmatrix}^{-1}$
$G_{k+1/2}^{(1)} \longleftarrow G_k^{(1)}$
$G_{k+1/2}^{(2)} \longleftarrow G_k^{(2)} - E_k Z_{21} F_k$
$H_{k+1/2}^{(1)} \longleftarrow H_k^{(1)} - F_k Z_{12} E_k$
$H_{k+1/2}^{(2)} \longleftarrow H_k^{(2)}$
$E_{k+1/2} \longleftarrow E_k Z_{11} E_k$
$F_{k+1/2} \longleftarrow F_k Z_{22} F_k$
Choose suitable nonsingular normalization factors $S_k^{(1)} \in \mathbb{R}^{n\times n}$, $S_k^{(2)} \in \mathbb{R}^{m\times m}$; e.g., compute QR factorizations of $\mathcal{H}_{k+1/2}^T$ and $\mathcal{G}_{k+1/2}^T$ and set $S_k^{(1)} \longleftarrow R_{\mathcal{G}_{k+1/2}^T}$,
$S_k^{(2)} \longleftarrow R_{\mathcal{H}_{k+1/2}^T}$
$G_{k+1}^{(1)} \longleftarrow S_k^{(1)} G_{k+1/2}^{(1)}$
$G_{k+1}^{(2)} \longleftarrow S_k^{(1)} G_{k+1/2}^{(2)}$
$H_{k+1}^{(1)} \longleftarrow S_k^{(2)} H_{k+1/2}^{(1)}$
$H_{k+1}^{(2)} \longleftarrow S_k^{(2)} H_{k+1/2}^{(2)}$
$E_{k+1} \longleftarrow S_k^{(1)} E_{k+1/2}$
$F_{k+1} \longleftarrow S_k^{(1)} F_{k+1/2}$
$k \longleftarrow k+1$
end while
return $U \longleftarrow \operatorname{null} \mathcal{H}_k$, $V \longleftarrow \operatorname{null} \mathcal{G}_k$

The iterated application of Remark 11.1 leads to an observation that is useful in the theoretical analysis.

Remark 11.2. Let $\mathcal{A}_k - z\mathcal{E}_k$ and $\mathcal{A}'_k - z\mathcal{E}'_k$, for $k = 0, 1, 2, \dots$ be two sequences obtained with Algorithm 17, the former with normalization factors $S_k^{(1)}, S_k^{(2)}$ and the latter with ${S'}_k^{(1)}, {S'}_k^{(2)}$. Then for each $k = 0, 1, 2, \dots$

$$\mathcal{A}'_k - z\mathcal{E}'_k = \begin{bmatrix} \Sigma_k^{(1)} & 0 \\ 0 & \Sigma_k^{(2)} \end{bmatrix} (\mathcal{A}_k - z\mathcal{E}_k),$$

with

$$\Sigma_k^{(i)} = {S'}_k^{(i)} \left(S_k^{(i)}\right)^{-1} {S'}_{k-1}^{(i)} \left(S_{k-1}^{(i)}\right)^{-1} \cdots {S'}_0^{(i)} \left(S_0^{(i)}\right)^{-1}, \quad i = 1, 2.$$

Convergence in the singular case

The following theorem tries to generalize the convergence results of the standard SDA.

Theorem 11.3 ([MP])**.** *Suppose that the normalization factors $S_k^{(i)}$ are chosen in Algorithm 17 such that $\|\mathcal{H}_k\|$ and $\|\mathcal{G}_k\|$ are bounded, and let*

$$U = \begin{bmatrix} U^{(1)} \\ U^{(2)} \end{bmatrix} \in \mathbb{R}^{(m+n)\times n}, \quad V = \begin{bmatrix} V^{(1)} \\ V^{(2)} \end{bmatrix} \in \mathbb{R}^{(m+n)\times m}$$

any two matrices spanning the canonical semi-d-stable and semi-d-unstable subspaces of the initial pencil (6.19)*. Then, the following implications hold.*

1. *if $U^{(1)}$ is nonsingular, then $\|E_k\| = O(2^{-k})$;*
2. *if $\|E_k\| = O(2^{-k})$, then $\|\mathcal{G}_k V\| = O(2^{-k})$;*
3. *if $V^{(2)}$ is nonsingular, then $\|F_k\| = O(2^{-k})$;*
4. *if $\|F_k\| = O(2^{-k})$, then $\|\mathcal{H}_k U\| = O(2^{-k})$;*

If the splitting is proper, the convergence is quadratic, and in particular in 3. and 4. the expression $O(2^{-k})$ can be replaced by $O(\nu^{2^k})$, where ν is the dominance factor (2.6) *of $\mathcal{H}$.*

Proof. We report here only the easier case in which there are no unimodular eigenvalues. The more involved case in which there are unimodular eigenvalues can be settled with minor modifications to the argument in [CCG+09, Section 5], which basically extends the same argument by considering the Jordan blocks relative to unimodular eigenvalues.

In this case, by Theorem 6.16,

$$\mathcal{A}_k U R_s = \mathcal{E}_k U R_s T_s^{2^k}, \tag{11.7}$$

$$\mathcal{A}_k V R_u T_u^{2^k} = \mathcal{E}_k V R_u T_u^{-2^k}, \tag{11.8}$$

for two invertible R_s, R_u and two matrices T_s and T_u, having respectively the stable eigenvalues and the unstable eigenvalues of the pencil.

By comparing the first block rows in the first equation, we get

$$E_k U^{(1)} R_s = \mathcal{G}_k U R_s T_s^{2^k} = O(\rho(T_s)^{2^k}),$$

so if $U^{(1)}$ is invertible E_k converges to zero itself and the first claim holds. By comparing the first block rows in the second equation, we get

$$\mathcal{G}_k V R_u = E_k V^{(1)} R_u T_u^{-2^k} = O(\rho(T_s)^{2^k} \rho(T_u)^{-2^k}) = O(\nu^{2^k}).$$

The other two claims follow by a similar argument on the second block row. □

Theorem 11.3 looks too weak for our uses: the boundedness of the H and G blocks can be achieved with a suitable normalization, but the hypothesis that $U^{(1)}$ and $V^{(2)}$ are invertible is exactly what we were trying to avoid in the first place. However, it is a necessary hypothesis: there are counterexamples as simple as

$$\mathcal{A} - z\mathcal{E} = \begin{bmatrix} \lambda & 0 \\ 0 & 1 \end{bmatrix} - z \begin{bmatrix} 1 & 0 \\ 0 & \lambda \end{bmatrix},$$

for $\lambda > 1$, for which $E_k = F_k = \lambda^{2^k}$, $\mathcal{G}_k = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and $\mathcal{H}_k = \begin{bmatrix} 0 & 1 \end{bmatrix}$, so $\operatorname{null} \mathcal{H}_k$ is the unstable space and $\operatorname{null} \mathcal{G}_k$ is the stable one, exactly the opposite of our goal.

In fact, there is a way to understand what happens in the framework of a more common algorithm, (orthogonal) power iteration [GVL96, Section 7.3]. Power iteration for a matrix $A \in \mathbb{R}^{n \times n}$ consists in starting from an initial subspace $\mathcal{W}_0 = \operatorname{Span} W_0$ and iterating the transformation $W_{k+1} = AW_k$, orthogonalizing the columns of W_k when needed in order to maintain a numerically stable basis for $\mathcal{W}_k = \operatorname{Span} W_k$. If $w = \dim \mathcal{W}_0$, and if the w dominant eigenvalues of A can be identified (i.e., if we order the eigenvalues of A so that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$, then we need $|\lambda_w| > |\lambda_{w+1}|$), then $\mathcal{W}_k$ converges linearly to the invariant subspace relative to $\lambda_1, \lambda_2, \ldots, \lambda_w$.

Theorem 11.4 ([MP])**.** *Suppose that both $\mathcal{E}_0$ and $\mathcal{A}_0$ are nonsingular, so that we can form a nonsingular $\mathcal{M} = \mathcal{E}_0^{-1}\mathcal{A}_0$. Then,*

- $\operatorname{null} \mathcal{H}_k$ *coincides with the subspace obtained by applying 2^k steps of power iteration to $\mathcal{M}^{-1}$, starting with $W_0 = [I_n \ 0]^T \in \mathbb{R}^{2n \times n}$.*
- $\operatorname{null} \mathcal{G}_k$ *coincides with the subspace obtained by applying 2^k steps of power iteration to $\mathcal{M}$, starting with $W_0 = [0 \ I_n]^T \in \mathbb{R}^{2n \times n}$;*

Proof. Let $U_k, V_k \in \mathbb{R}^{2n \times n}$ be such that $\operatorname{Span} U_k = \operatorname{null} \mathcal{H}_k$, $\operatorname{Span} V_k = \operatorname{null} \mathcal{G}_k$, and let

$$U_k = \begin{bmatrix} U_k^{(1)} \\ U_k^{(2)} \end{bmatrix}.$$

Then,

$$\begin{aligned} \mathcal{M}^{2^k} U_k =& (\mathcal{E}_0^{-1}\mathcal{A}_0)^{2^k} U_k = (\mathcal{E}_k^{-1}\mathcal{A}_k) U_k = \begin{bmatrix} G_k^{(1)} & G_k^{(2)} \\ 0 & F_k \end{bmatrix}^{-1} \begin{bmatrix} E_k & 0 \\ H_k^{(1)} & H_k^{(2)} \end{bmatrix} U_k \\ =& \begin{bmatrix} G_k^{(1)} & G_k^{(2)} \\ 0 & F_k \end{bmatrix}^{-1} \begin{bmatrix} E_k U_k^{(1)} \\ 0 \end{bmatrix} = \begin{bmatrix} \left(G_k^{(1)}\right)^{-1} E_k U_k^{(1)} \\ 0 \end{bmatrix} \subseteq \operatorname{Span} \begin{bmatrix} I_n \\ 0 \end{bmatrix}. \end{aligned}$$

A similar argument starting from $\mathcal{M}^{-2^k} V_0$ yields the second part. □

This fact can be used to provide a more direct proof of Theorem 11.3, if we consider carefully the behavior of power iteration when $\lambda_w = \lambda_{w+1}$ in presence of nontrivial Jordan blocks.

It is well established [Wil88] that the presence of rounding errors is usually beneficial for power iteration, as it often makes the method converge to the correct subspace even if the starting subspace is deficient along some of the leading eigendirections. Something similar happens with SDA: when $U^{(1)}$ is singular, the starting subspace $\mathcal{U}_0$ and U are not in generic position (as can be checked by computing $U^T\mathcal{U}_0$), but still in many cases this exact orthogonality is lost in computer arithmetic. The computational results in Section 11.3

suggest that this is what happens indeed in most cases. In view of Theorem 11.3, convergence to the correct subspace happens if E_k and F_k converge to zero, so this provides a practical criterion to check that this rounding error magic has happened.

The symplectic case

The most important application of SDA is the solution of continuous-time and discrete-time algebraic Riccati equations. These applications result in a starting pencil which is symplectic, i.e., $E_0 = F_0^T$, $G_0 = G_0^T$, $H_0 = H_0^T$. One can verify directly that in this case all the pencils generated by the successive steps of SDA are symplectic, i.e., at each step k we have $E_k^* = F_k$, $G_k = G_k^*$, $H_k = H_k^*$. The implementation in Algorithm 8 can be slightly simplified, since there is no need to compute E_{k+1} and F_{k+1} separately, nor to invert both $I_n - G_k H_k$ and $I_m - H_k G_k$, as the second matrix of both pairs is the transposed of the first.

Traditional SDA preserves this structure, and it is a natural question to ask whether this happens for Algorithm 17 too, under a suitable choice of the normalization. We have a partial positive result, which shows that the correct generalization of the Hermitianity of G_k and H_k is preserved.

Theorem 11.5 ([MP]). *Along the iterates of Algorithm 17, $\mathcal{H}_k^T$ and $\mathcal{G}_k^T$ are Lagrangian subspaces, i.e., $H_k^{(2)} {H_k^{(1)}}^T = H_k^{(1)} {H_k^{(2)}}^T$ and $G_k^{(2)} {G_k^{(1)}}^T = G_k^{(1)} {G_k^{(2)}}^T$. Moreover, $H_k^{(2)} E_k^T = F_k {G_k^{(1)}}^T$.*

Proof. By Remark 11.2, Algorithm 17 differs from traditional SDA (Algorithm 8) only by a normalization factor in the form

$$\begin{bmatrix} \Sigma_k^{(1)} & 0 \\ 0 & \Sigma_k^{(2)} \end{bmatrix}.$$

Direct inspection of the blocks shows that these factors must be $\Sigma_k^{(1)} = {G_k^{(1)}}^{-1}$ and $\Sigma_k^{(2)} = {H_k^{(2)}}^{-1}$. Since in the traditional SDA the E_k block is the transpose of the F_k block, we get $\left({G_k^{(1)}}^{-1} E_k\right)^T = {H_k^{(2)}}^{-1} F_k$, that is, $H_k^{(2)} E_k^T = F_k {G_k^{(1)}}^T$. Moreover, notice that this normalization corresponds to choosing new bases for the subspaces $\mathcal{H}_k^T$ and $\mathcal{G}_k^T$, without changing the subspaces themselves. It is easy to check using the iterates of traditional SDA that the subspaces spanned by $\begin{bmatrix} H_k & I_m \end{bmatrix}^T$ and $\begin{bmatrix} I_n & G_k \end{bmatrix}^T$ are Lagrangian, as this corresponds to H_k and G_k being Hermitian. Since Lagrangianness is a property of the subspace, and not of the choice of the basis, this also holds for Algorithm 17. □

This fact can be exploited in Algorithm 17 to perform a faster (and structured) matrix inversion in (11.3). Indeed, one sees that

$$\begin{bmatrix} G^{(1)} & G^{(2)} \\ H^{(1)} & H^{(2)} \end{bmatrix}^{-1} = \mathcal{J} \begin{bmatrix} G^{(1)} & G^{(2)} \\ H^{(1)} & H^{(2)} \end{bmatrix}^T \begin{bmatrix} 0 & R^{-1} \\ R^{-T} & 0 \end{bmatrix},$$

with $R = H^{(1)} {G^{(2)}}^T - H^{(2)} {G^{(1)}}^T$. The final step can be simplified as well, since $\operatorname{null} \mathcal{H}_k = \mathcal{J} \mathcal{H}_k^T$.

On the other hand, the normalization choice suggested in Algorithm 17 (QR factorization of $\mathcal{H}_{k+1/2}^T$ and $\mathcal{G}_{k+1/2}^T$) does not guarantee that $E_k = F_k^T$, which would be useful to reduce the computational costs and ensure structure preservation. Ideally, we would like the following conditions to hold at the same time.

P1 $F_k = E_k^T$ for each k

P2 $\mathcal{H}_k$ and $\mathcal{G}_k$ have orthonormal rows;

In fact, the following negative results can be proved.

Theorem 11.6 ([MP]). *There are no normalization factors $S_k^{(1)}$ and $S_k^{(2)}$ in Algorithm 17 ensuring that for all possible starting pencils* (6.19) *P1 and P2 hold.*

Proof. The condition $F_k = E_k^T$ means that we must choose $S_k^{(2)} = {S_k^{(1)}}^T = S_k$ at each k. If this and P2 held at the same time, then $S_k \mathcal{H}_{k+1/2} \mathcal{H}_{k+1/2}^T S_k^T = S_k^T \mathcal{G}_{k+1/2} \mathcal{G}_{k+1/2}^T S_k = I$, which would imply that $\mathcal{H}_{k+1/2}\mathcal{H}_{k+1/2}^T = S_k^{-1} S_k^{-T}$ and $\mathcal{G}_{k+1/2}\mathcal{G}_{k+1/2}^T = S_k^{-T} S_k^{-1}$ have the same eigenvalues, and this does not hold in general. □

So we cannot have P1 and P2 at the same time. Asking for P1 yields traditional SDA, or in general algorithms that tend to have convergence problems in the case in which $U^{(1)}$ and $V^{(2)}$ are close to singular. Asking P2 yields Algorithm 17, which performs better for this class of problems, according to our experiments.

11.3 Numerical experiments

The experiments in the benchmark suites `carex` [BLM95a] and `darex` [BLM95b] are designed for Riccati equations, i.e., in our language, they assume nonsingular $U^{(1)}$ and $V^{(2)}$ blocks from the beginning. In order to generate examples with numerically singular $U^{(1)}$ and $V^{(2)}$, we chose problems in `carex` and considered the optimal control problem with Hamiltonian $-H$ instead of H, with a sign change. This has the effect of switching the c-stable and c-unstable subspaces, that is, swapping $U^{(1)}$, $V^{(2)}$ with $V^{(1)}$, $U^{(2)}$. In this way the original `carex` problems with a singular solution X are transformed into problems for which $V^{(2)}$ is numerically singular. Several of these problems, which are nonetheless *bona fide* optimal control problems, lead to convergence failure or large errors in SDA.

For each problem in `carex` (after the sign switch of the Hamiltonian), we computed the canonical semi-stable and semi-unstable invariant subspaces U and V with both SDA and gSDA. For each of them, we tested as an error metric the distance from invariance in the gap metric, i.e.,

$$\theta(\operatorname{Span} U, \operatorname{Span} \mathcal{H}U),$$

where, according to [GLR06], the distance between subspaces is given by

$$\theta(\mathcal{L}, \mathcal{M}) = \|P_{\mathcal{L}} - P_{\mathcal{M}}\|,$$

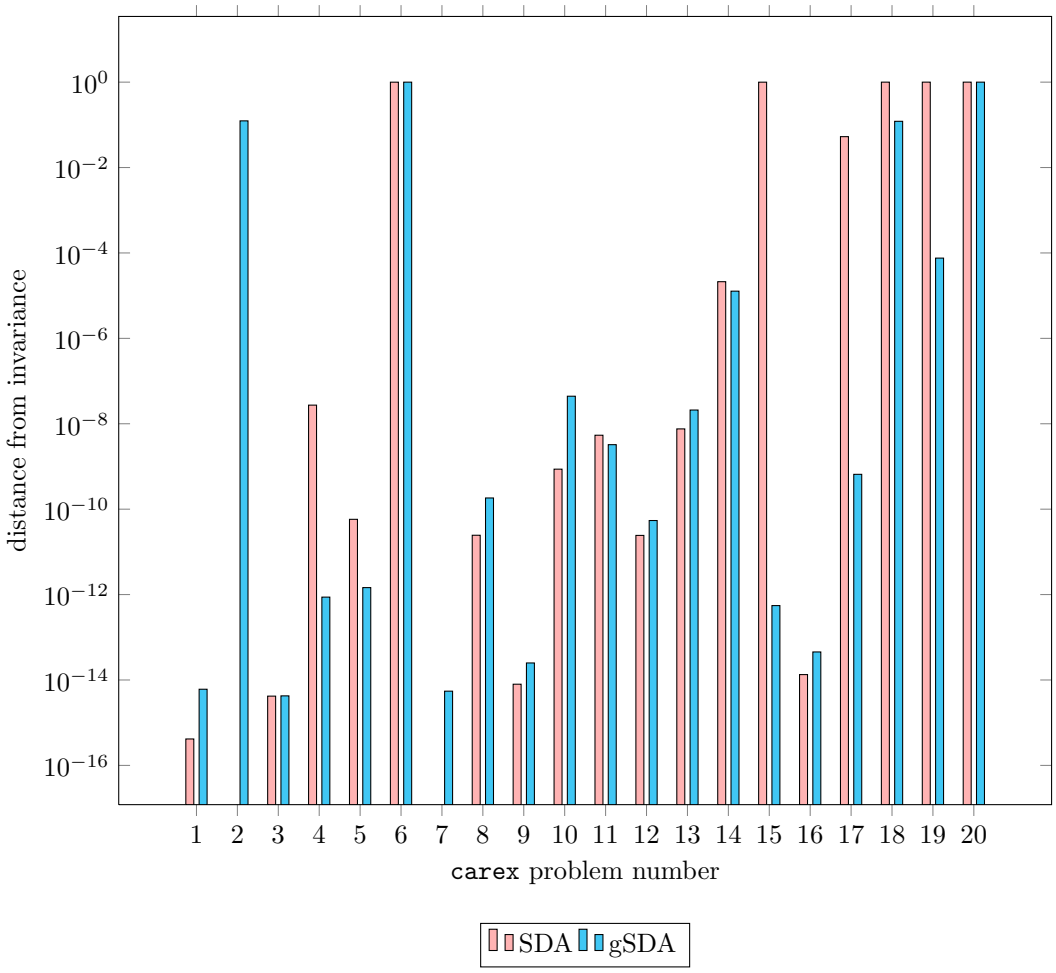
with $P_{\mathcal{L}}$ and $P_{\mathcal{M}}$ the orthogonal projectors on $\mathcal{L}$ and $\mathcal{M}$ respectively, and $\|\cdot\|$ denoting the matrix norm induced by the Euclidean norm.

Figure 11.1 reports the plot of

$$\max(\theta(\operatorname{Span} U, \operatorname{Span} \mathcal{H}U), \theta(\operatorname{Span} V, \operatorname{Span} \mathcal{H}V))$$

for all problems in the `carex` benchmark set. A Cayley transform with parameter $\gamma = 1$ was chosen to transform the pencils to symplectic SSF-I, the only exception being problem 15 for which $\gamma = 2$ (since the SSF-I for the pencil with $\gamma = 1$ does not exist). On problem 2 and 7, SDA failed to converge, while gSDA converged, though with bad accuracy on problem 2. On several problems in the set gSDA was significantly more accurate than its counterpart;

Figure 11.1: Invariant subspace residual for the examples in `carex`



on the other hand, in some problems SDA resulted in a marginally better result (no more than 1 significant digit, two only for problem 10).

According to [BLM95a, table 1], `carex` problems whose solution $\hat{X}$ is exactly singular (and thus $V^{(2)}$ is singular and we expect ill-conditioning in our modified problems) are 2, 6, 15, 17 and 18. The solution $\hat{X}$ is severely ill-conditioned in problems 7 and 20.

Convergence history

To provide insight on the reasons for the instabilities remaining in the two methods, it is interesting to report the convergence history of the two methods on some of the problems where the block $V^{(2)}$ is singular. In Tables 11.1 and 11.2 we report the norms of E_k for both method at each iterations. For gSDA, the norm of F_k is reported as well ($E_k = F_k^T$ holds for SDA, thus the two norms coincide). For SDA, the column labeled cond. contains

$$\max(\mathrm{cond}(I - G_k H_k), \mathrm{cond}(I - H_k G_k));$$

for gSDA, it contains

$$\max(\mathrm{cond}(R_{\mathcal{G}_k^T}), \mathrm{cond}(R_{\mathcal{H}_k^T}));$$

i.e., in both cases the maximum condition number among the matrices to invert at each step.

On problem 2, numerical errors in SDA do not alter the singularity of the $V^{(2)}$ block, thus the algorithm diverges; $\mathrm{null}\,\mathcal{H}_k$ contains a basis for the unstable subspace at each step, thus the repeated doubling takes E_k to infinity instead of to 0. In gSDA, a similar behavior appears until in step 5 the inversion of an ill-conditioned matrix destroys completely the initial data. The method eventually converges, but to a solution bearing little resemblance to the correct one. Similar behavior arises in problem 6, where in both methods the growth of E_k and F_k leads to ill-conditioning in an intermediate matrix and complete loss of accuracy. On problems 15 and 17, the growth in gSDA is much milder than the one in SDA, thus the former can obtain a meaningful result.

11.4 Conclusions and research lines

In this chapter we have proposed a generalization of the SDA approach, which is based on a work in preparation [MP]. It represents a compromise between row orthogonality of the CS-AB solution, as in the inverse-free disk iteration, and symplectic structure preservation and numerical efficiency, as in the customary SDA. The generalized framework allows to construct different variants, based on different choices of the normalization factors $S_k^{(1)}$ and $S_k^{(2)}$.

Many of the problems included in the benchmark set `carex` are challenging for most algorithms to solve Riccati equations; although our variant of SDA cannot handle all of them, it obtains much better accuracy on a large subset. In some cases, a temporary growth of the matrices E_k and F_k during the intermediate steps is detrimental for the accuracy. Though this algorithm cannot be considered a definitive choice for all small- to medium-size control problems, it represents an improvement over the customary SDA and suggests new generalizations and research directions.

Other approaches to deal with the singularity of the $U^{(1)}$ and $V^{(2)}$ blocks could be considered. An idea which we plan to analyze is generating a random orthogonal and symplectic matrix S and looking for the canonical semi-stable subspace of $S\mathcal{H}S^{-1}$ instead of $\mathcal{H}$. This would ensure, at least with high probability, that the power iteration underlying SDA (see Theorem 11.4) does not encounter problems due to rank deficiency in a specific eigendirection. This idea is similar to the randomized URV decomposition approach suggested in [DDH07] for spectral division algorithms.

Table 11.1: Convergence history for SDA and gSDA

Problem 2 from `carex`					
	SDA		gSDA		
k	$\|E_k\|$	cond.	$\|E_k\|$	$\|F_k\|$	cond.
1	$4.6 \times 10^{+01}$	$2.2 \times 10^{+00}$	$3.6 \times 10^{+01}$	$1.2 \times 10^{+01}$	$5.5 \times 10^{+00}$
2	$4.1 \times 10^{+02}$	$2.2 \times 10^{+00}$	$3.2 \times 10^{+02}$	$1.1 \times 10^{+02}$	$1.0 \times 10^{+00}$
3	$3.4 \times 10^{+04}$	$2.2 \times 10^{+00}$	$2.6 \times 10^{+04}$	$8.9 \times 10^{+03}$	$1.0 \times 10^{+00}$
4	$2.2 \times 10^{+08}$	$1.3 \times 10^{+00}$	$8.9 \times 10^{+07}$	$4.3 \times 10^{+07}$	$1.9 \times 10^{+00}$
5	$8.5 \times 10^{+15}$	$1.6 \times 10^{+16}$	$4.3 \times 10^{+00}$	$3.4 \times 10^{+00}$	$9.7 \times 10^{+14}$
6	$4.5 \times 10^{+01}$	Inf	$1.6 \times 10^{+00}$	1.9×10^{-15}	$1.6 \times 10^{+01}$
7	breakdown		4.0×10^{-01}	3.1×10^{-46}	$1.0 \times 10^{+00}$
8			2.4×10^{-02}	3.8×10^{-107}	$1.0 \times 10^{+00}$
9			8.5×10^{-05}	2.2×10^{-228}	$1.0 \times 10^{+00}$
10			1.1×10^{-09}	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
11			1.8×10^{-19}	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
12			4.7×10^{-39}	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
13			3.3×10^{-78}	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
14			1.6×10^{-156}	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
15			4.1×10^{-313}	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
16			$0.0 \times 10^{+00}$	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
				convergence	

Problem 6 from `carex`					
	SDA		gSDA		
k	$\|E_k\|$	cond.	$\|E_k\|$	$\|F_k\|$	cond.
1	$4.3 \times 10^{+02}$	$2.3 \times 10^{+05}$	$5.2 \times 10^{+01}$	$5.3 \times 10^{+01}$	$9.0 \times 10^{+03}$
2	$1.8 \times 10^{+03}$	$3.1 \times 10^{+06}$	$7.5 \times 10^{+01}$	$3.3 \times 10^{+02}$	$1.4 \times 10^{+04}$
3	$8.8 \times 10^{+04}$	$3.0 \times 10^{+10}$	$1.1 \times 10^{+03}$	$7.9 \times 10^{+04}$	$4.7 \times 10^{+06}$
4	$2.5 \times 10^{+10}$	$3.5 \times 10^{+20}$	$5.3 \times 10^{+02}$	$4.7 \times 10^{+09}$	$1.4 \times 10^{+15}$
5	$3.0 \times 10^{+20}$	$3.6 \times 10^{+43}$	$1.4 \times 10^{+03}$	$8.8 \times 10^{+16}$	$2.2 \times 10^{+31}$
6	$5.2 \times 10^{+16}$	$1.2 \times 10^{+49}$	$1.3 \times 10^{+03}$	$2.5 \times 10^{+17}$	$3.1 \times 10^{+31}$
7	$5.6 \times 10^{+13}$	$1.3 \times 10^{+47}$	$8.4 \times 10^{+04}$	$4.1 \times 10^{+17}$	$1.3 \times 10^{+34}$
8	$4.3 \times 10^{+13}$	$1.9 \times 10^{+44}$	$2.6 \times 10^{+05}$	$1.2 \times 10^{+15}$	$6.4 \times 10^{+32}$
9	$1.5 \times 10^{+09}$	$2.7 \times 10^{+41}$	$3.4 \times 10^{+04}$	$2.0 \times 10^{+16}$	$6.9 \times 10^{+30}$
10	2.7×10^{-01}	$1.2 \times 10^{+40}$	$1.1 \times 10^{+03}$	$2.3 \times 10^{+17}$	$5.7 \times 10^{+31}$
11	9.5×10^{-22}	$1.2 \times 10^{+40}$	9.1×10^{-01}	$5.7 \times 10^{+18}$	$1.4 \times 10^{+31}$
12	1.2×10^{-62}	$1.2 \times 10^{+40}$	6.8×10^{-07}	$6.8 \times 10^{+21}$	$4.3 \times 10^{+22}$
13	1.8×10^{-144}	$1.2 \times 10^{+40}$	3.9×10^{-19}	$1{,}0.0 \times 10^{+27}$	$2.6 \times 10^{+14}$
14	4.1×10^{-308}	$1.2 \times 10^{+40}$	1.3×10^{-43}	$2.1 \times 10^{+40}$	$1.8 \times 10^{+02}$
15	$0.0 \times 10^{+00}$	$1.2 \times 10^{+40}$	1.3×10^{-92}	$9.8 \times 10^{+64}$	$1.0 \times 10^{+00}$
16	convergence		1.4×10^{-190}	$2.1 \times 10^{+114}$	$1.0 \times 10^{+00}$
17			$0.0 \times 10^{+00}$	$9.2 \times 10^{+212}$	$1.0 \times 10^{+00}$
				convergence	

Table 11.2: Convergence history for SDA and gSDA

Problem 15 from `carex`

	SDA		gSDA		
k	$\|E_k\|$	cond.	$\|E_k\|$	$\|F_k\|$	cond.
1	$1.0 \times 10^{+01}$	$1.0 \times 10^{+02}$	$1.5 \times 10^{+00}$	$9.0 \times 10^{+00}$	$4.0 \times 10^{+01}$
2	$8.1 \times 10^{+01}$	$8.7 \times 10^{+01}$	8.3×10^{-01}	$8.1 \times 10^{+01}$	$8.4 \times 10^{+01}$
3	$6.6 \times 10^{+03}$	$1.4 \times 10^{+02}$	1.1×10^{-01}	$6.6 \times 10^{+03}$	$6.7 \times 10^{+03}$
4	$4.3 \times 10^{+07}$	$1.2 \times 10^{+13}$	6.6×10^{-04}	$4.3 \times 10^{+07}$	$4.3 \times 10^{+07}$
5	$2.0 \times 10^{+02}$	$3.0 \times 10^{+19}$	1.4×10^{-08}	$7.4 \times 10^{+01}$	$1.7 \times 10^{+02}$
6	7.0×10^{-11}	$1.6 \times 10^{+18}$	3.8×10^{-18}	1.9×10^{-16}	$1.0 \times 10^{+00}$
7	1.5×10^{-27}	$1.6 \times 10^{+18}$	2.9×10^{-37}	1.4×10^{-35}	$1.0 \times 10^{+00}$
8	2.8×10^{-60}	$1.6 \times 10^{+18}$	1.6×10^{-75}	7.7×10^{-74}	$1.0 \times 10^{+00}$
9	7.7×10^{-126}	$1.6 \times 10^{+18}$	4.8×10^{-152}	2.3×10^{-150}	$1.0 \times 10^{+00}$
10	5.3×10^{-257}	$1.6 \times 10^{+18}$	4.5×10^{-305}	2.2×10^{-303}	$1.0 \times 10^{+00}$
11	$0.0 \times 10^{+00}$	$1.6 \times 10^{+18}$	$0.0 \times 10^{+00}$	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
	convergence			convergence	

Problem 17 from `carex`

	SDA		gSDA		
k	$\|E_k\|$	cond.	$\|E_k\|$	$\|F_k\|$	cond.
1	$1.8 \times 10^{+02}$	$4.4 \times 10^{+02}$	$7.9 \times 10^{+01}$	$7.9 \times 10^{+01}$	$2.1 \times 10^{+01}$
2	$6.2 \times 10^{+03}$	$1.5 \times 10^{+06}$	$8.4 \times 10^{+02}$	$8.4 \times 10^{+02}$	$1.2 \times 10^{+03}$
3	$7.1 \times 10^{+05}$	$8.1 \times 10^{+10}$	$8.9 \times 10^{+03}$	$8.9 \times 10^{+03}$	$8.6 \times 10^{+05}$
4	$8.0 \times 10^{+07}$	$8.1 \times 10^{+15}$	$2.3 \times 10^{+04}$	$2.3 \times 10^{+04}$	$4.2 \times 10^{+08}$
5	$5.3 \times 10^{+08}$	$1.2 \times 10^{+19}$	$8.8 \times 10^{+03}$	$8.8 \times 10^{+03}$	$2.4 \times 10^{+09}$
6	$4.7 \times 10^{+07}$	$9.4 \times 10^{+18}$	$8.0 \times 10^{+02}$	$8.0 \times 10^{+02}$	$8.4 \times 10^{+07}$
7	$8.3 \times 10^{+04}$	$1.3 \times 10^{+19}$	$6.7 \times 10^{+00}$	$6.7 \times 10^{+00}$	$9.1 \times 10^{+03}$
8	2.5×10^{-01}	$3.7 \times 10^{+19}$	4.6×10^{-04}	4.6×10^{-04}	$1.0 \times 10^{+00}$
9	3.3×10^{-12}	$3.7 \times 10^{+19}$	2.2×10^{-12}	2.2×10^{-12}	$1.0 \times 10^{+00}$
10	5.4×10^{-34}	$3.7 \times 10^{+19}$	4.9×10^{-29}	4.9×10^{-29}	$1.0 \times 10^{+00}$
11	1.4×10^{-77}	$3.7 \times 10^{+19}$	2.5×10^{-62}	2.5×10^{-62}	$1.0 \times 10^{+00}$
12	9.8×10^{-165}	$3.7 \times 10^{+19}$	6.3×10^{-129}	6.3×10^{-129}	$1.0 \times 10^{+00}$
13	$0.0 \times 10^{+00}$	$3.7 \times 10^{+19}$	4.2×10^{-262}	4.2×10^{-262}	$1.0 \times 10^{+00}$
14	convergence		$0.0 \times 10^{+00}$	$0.0 \times 10^{+00}$	$1.0 \times 10^{+00}$
				convergence	

Part IV

Matrix geometric means

Chapter 12

An effective matrix geometric mean

12.1 Introduction

In several contexts, it is natural to generalize the geometric mean of two positive real numbers $a \# b := \sqrt{ab}$ to real symmetric positive definite $n \times n$ matrices using (1.15) Several papers, e.g. [BH06a, BH06b, Moa06], and a chapter of the book [Bha07], are devoted to studying the geometry of the cone of positive definite matrices $\mathbb{P}^n$ endowed with the Riemannian metric defined by

$$ds = \left\| A^{-1/2} dA A^{-1/2} \right\|,$$

where $\|B\| = \sqrt{\sum_{i,j} |b_{i,j}|^2}$ denotes the Frobenius norm. The distance induced by this metric is

$$d(A, B) = \left\| \log(A^{-1/2} B A^{-1/2}) \right\| \tag{12.1}$$

It turns out that on this manifold the geodesic joining X and Y has equation

$$\gamma(t) = X^{1/2}(X^{-1/2} Y X^{-1/2})^t X^{1/2} = X(X^{-1}Y)^t =: X \#_t Y, \qquad t \in [0, 1],$$

and thus $A \# B$ is the midpoint of the geodesic joining A and B. An analysis of numerical methods for computing the geometric mean of two matrices is carried out in [IM09].

It is less clear how to define the geometric mean of more than two matrices. In the seminal work [ALM04], Ando, Li and Mathias list ten properties that a "good" matrix geometric mean should satisfy, and show that several natural approaches based on generalization of formulas working for the scalar case, or for the case of two matrices, do not work well. They propose a new definition of mean of k matrices satisfying all the requested properties. We refer to this mean as to the Ando–Li–Mathias mean, or shortly ALM mean.

The ALM mean is the limit of a recursive iteration process where at each step of the iteration k geometric means of $k - 1$ matrices must be computed. One of the main drawback of this iteration is its linear convergence. In fact, the large number of iterations needed to approximate each geometric mean at all the recursive steps makes it quite expensive to actually compute the ALM mean with this algorithm. Moreover, no other algorithms endowed with a higher efficiency are known.

A class of geometric means satisfying the Ando, Li, Mathias requirements has been introduced in [Lim08]. These means are defined in terms of the solution of certain matrix equations. This approach provides interesting theoretical properties concerning the means but no effective tools for their computation.

In this chapter, we propose a new matrix geometric mean satisfying the ten properties of Ando, Li and Mathias. Like the ALM mean, our mean is defined as the limit of an iteration process with the relevant difference that convergence is superlinear with order of convergence at least three. This property makes it much less expensive to compute this geometric mean since the number of iterations required to reach a high accuracy is dropped down to just a few ones.

The iteration on which our mean is based has a simple geometrical interpretation. In the case $k = 3$, given the positive definite matrices $A = A^{(0)}$, $B = B^{(0)}$, $C = C^{(0)}$, we generate three matrix sequences $A^{(m)}, B^{(m)}, C^{(m)}$. For each $m \geq 0$, the matrix $A^{(m+1)}$ is chosen along the geodesic which connects $A^{(m)}$ to the midpoint of the geodesic connecting $B^{(m)}$ and $C^{(m)}$. The matrices $B^{(m+1)}$ and $C^{(m+1)}$ are defined similarly, by cycling the letters. In the case of the Euclidean geometry, just one step of the iteration provides the value of the limit, i.e., the centroid of the triangle with vertices A, B, C. This is because in a triangle the medians intersect each other at $\frac{2}{3}$ of their length. In the different geometry of the cone of positive definite matrices, the geodesics which play the role of the medians might even not intersect each other.

In the case of k matrices $A_1, A_2, \ldots, A_k$, the matrix $A_i^{(m+1)}$ is chosen along the geodesic which connects $A_i^{(m)}$ with the geometric mean of the remaining matrices, at distance $\frac{k}{(k+1)}$ from $A_i^{(m)}$. In the customary geometry, this point is the common intersection point of all the "medians" of the k-dimensional simplex with vertices $A_i^{(m)}$, $i = 1, \ldots, k$. We prove that the sequences $\{A_i^{(m)}\}_m$, $i = 1, \ldots, k$, converge to a common limit $\overline{A}$ with order of convergence at least 3. The limit $\overline{A}$ is our definition of geometric mean of $A_1, \ldots, A_k$.

It is interesting to point out that our mean and the ALM mean of k matrices can be viewed as two specific instances of a class of more general means depending on $k - 1$ parameters $s_i \in [0, 1]$, $i = 1, \ldots, k - 1$. All the means of this class satisfy the requirements of Ando, Li and Mathias, moreover, the ALM mean is obtained with $\boldsymbol{s} = (\frac{1}{2}, 1, 1, \ldots, 1)$, for $\boldsymbol{s} = (s_i)$, while our mean is obtained with $\boldsymbol{s} = (\frac{k-1}{k}, \frac{k-2}{k-1}, \ldots, \frac{1}{2})$. The new mean is the only one in this class for which the matrix sequences generated at each recursive step converge superlinearly.

The article is structured as follows. After this introduction, in Section 12.2 we present the ten Ando–Li–Mathias properties and briefly describe the ALM mean; then, in Section 12.3, we propose our new definition of a matrix geometric mean and prove some of its properties by giving also a geometrical interpretation; in Section 12.4 we provide a generalization which includes the ALM mean and our mean as two special cases. Finally, in Section 12.5 we present some numerical experiments of explicit computations involving this means concerning some problems from Physics. It turns out that, in the case of six matrices, the speed up reached by our approach with respect to the ALM mean is by a factor greater than 200.

12.2 Known results

We recall that throughout this part we use the positive semidefinite ordering.

Ando–Li–Mathias properties for a matrix geometric mean

Ando, Li and Mathias [ALM04] proposed the following list of properties that a "good" geometric mean $G(\cdot)$ of three matrices should satisfy.

P1 Consistency with scalars. If A, B, C commute then $G(A, B, C) = (ABC)^{1/3}$.

P2 Joint homogeneity. $G(\alpha A, \beta B, \gamma C) = (\alpha\beta\gamma)^{1/3} G(A, B, C)$.

P3 Permutation invariance. For any permutation $\pi(A, B, C)$ of A, B, C it holds $G(A, B, C) = G(\pi(A, B, C))$.

P4 Monotonicity. If $A \geq A'$, $B \geq B'$, $C \geq C'$, then $G(A, B, C) \geq G(A', B', C')$.

P5 Continuity from above. If A_n, B_n, C_n are monotonic decreasing sequences converging to A, B, C, respectively, then $G(A_n, B_n, C_n)$ converges to $G(A, B, C)$.

P6 Congruence invariance. For any nonsingular S, $G(S^*AS, S^*BS, S^*CS) = S^*G(A, B, C)S$.

P7 Joint concavity. If $A = \lambda A_1 + (1-\lambda)A_2$, $B = \lambda B_1 + (1-\lambda)B_2$, $C = \lambda C_1 + (1-\lambda)C_2$, then $G(A, B, C) \geq \lambda G(A_1, B_1, C_1) + (1-\lambda)G(A_2, B_2, C_2)$.

P8 Self-duality. $G(A, B, C)^{-1} = G(A^{-1}, B^{-1}, C^{-1})$.

P9 Determinant identity. $\det G(A, B, C) = (\det A \det B \det C)^{1/3}$.

P10 Arithmetic–geometric–harmonic mean inequality:

$$\frac{A+B+C}{3} \geq G(A, B, C) \geq \left(\frac{A^{-1}+B^{-1}+C^{-1}}{3}\right)^{-1}.$$

Moreover, it is proved in [ALM04] that P5 and P10 are consequences of the others. Notice that all these properties can be easily generalized to the mean of any number of matrices. We call a *geometric mean* of three or more matrices any map $G(\cdot)$ satisfying P1–P10 or their analogous for a number $k \neq 3$ of entries.

The Ando–Li–Mathias mean

Here and hereafter, we use the following notation. We denote by $G_2(A, B)$ the usual geometric mean $A \# B$ and, given the k-tuple $(A_1, \ldots, A_k)$, we define

$$\mathcal{Z}_i(A_1, \ldots, A_k) = (A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_k), \qquad i = 1, \ldots, k,$$

that is, $\mathcal{Z}_i$ is the operator that drops the i-th term of a k-tuple.

In [ALM04], Ando, Li and Mathias note that the previously proposed definitions of means of more than two matrices do not satisfy all the properties P1–P10, and propose a new definition that fulfills all of them. Their mean is defined inductively on the number of arguments k.

Given $A_1, \ldots, A_k$ positive definite, and given the definition of a geometric mean $G_{k-1}(\cdot)$ of $k-1$ matrices, they set $A_i^{(0)} = A_i$, $i = 1, \ldots, k$, and define for $r \geq 0$

$$A_i^{(r+1)} := G_{k-1}(\mathcal{Z}_i(A_1^{(r)}, \ldots, A_k^{(r)})), \qquad i = 1, \ldots, k. \tag{12.2}$$

For $k = 3$, the iteration reads

$$\begin{bmatrix} A^{(r+1)} \\ B^{(r+1)} \\ C^{(r+1)} \end{bmatrix} = \begin{bmatrix} G_2(B^{(r)}, C^{(r)}) \\ G_2(A^{(r)}, C^{(r)}) \\ G_2(A^{(r)}, B^{(r)}) \end{bmatrix}.$$

Ando, Li and Mathias show that the k sequences $(A_i^{(r)})_{r=1}^{\infty}$ converge to the same matrix $\tilde{A}$, and finally define $G_k(A_1, \ldots, A_k) = \tilde{A}$. In the following, we denote by $G(\cdot)$ the Ando–Li–Mathias mean, dropping the subscript k when not essential.

An additional property of the Ando–Li–Mathias mean which turns out to be important in the convergence proof is the following. Recall that $\rho(X)$ denotes the spectral radius of X, and let

$$R(A,B) := \max(\rho(A^{-1}B), \rho(B^{-1}A)).$$

This function is a *multiplicative metric*, that is, we have $R(A,B) \geq 1$ with equality if and only if $A = B$, and

$$R(A,C) \leq R(A,B)R(B,C).$$

The additional property is

P11 For each $k \geq 2$, and for each pair of sequences $(A_1,\ldots,A_k)$, $(B_1,\ldots,B_k)$ it holds

$$R\left(G(A_1,\ldots,A_k), G(B_1,\ldots,B_k)\right) \leq \left(\prod_{i=1}^{k} R(A_i,B_i)\right)^{1/k}.$$

12.3 A new matrix geometric mean

Definition

We define now for each $k \geq 2$ a new mean $\overline{G}_k(\cdot)$ of k matrices satisfying P1–P11. Let $\overline{G}_2(A,B) = A \# B$, and suppose that the mean has already been defined for up to $k-1$ matrices. Let us denote shortly $T_i^{(r)} = \overline{G}_{k-1}(\mathcal{Z}_i(\overline{A}_1^{(r)},\ldots,\overline{A}_k^{(r)}))$ and define $\overline{A}_i^{(r+1)}$ for $i = 1,\ldots,k$ as

$$\overline{A}_i^{(r+1)} := \overline{G}_k(\overline{A}_i^{(r)}, \underbrace{T_i^{(r)}, T_i^{(r)}, \ldots, T_i^{(r)}}_{k-1 \text{ times}}), \tag{12.3}$$

with $\overline{A}_i^{(0)} = A_i$ for all i. Notice that apparently this needs the mean $\overline{G}_k(\cdot)$ to be already defined; in fact, in the special case in which $k-1$ of the k arguments are coincident, the properties P1 and P6 alone allow one to determine the common value of any geometric mean:

$$\begin{aligned} G(X,Y,Y,\ldots,Y) =& X^{1/2} G(I, X^{-1/2}YX^{-1/2}, \ldots, X^{-1/2}YX^{-1/2}) X^{1/2} \\ =& X^{1/2}(X^{-1/2}YX^{-1/2})^{\frac{k-1}{k}} X^{1/2} = X \#_{\frac{k-1}{k}} Y. \end{aligned}$$

Thus we can use this simpler expression directly in (12.3) and set

$$\overline{A}_i^{(r+1)} = \overline{A}_i^{(r)} \#_{\frac{k-1}{k}} T_i^{(r)}. \tag{12.4}$$

In sections 12.3 and 12.3, we prove that the k sequences $(\overline{A}_i^{(r)})_{r=1}^{\infty}$ converge to a common limit $\overline{A}$ with order of convergence at least three, and this enables us to define $\overline{G}_k(A_1,\ldots,A_k) := \overline{A}$. In the following, we drop the index k from $\overline{G}_k(\cdot)$ when it can be easily inferred from the context.

Geometrical interpretation

In [BH06a], an interesting geometrical interpretation of the Ando–Li–Mathias mean is proposed for $k = 3$. We propose an interpretation of the new mean $\overline{G}(\cdot)$ in the same spirit.

For $k = 3$, the iteration defining $\overline{G}(\cdot)$ reads

$$\begin{bmatrix} \overline{A}^{(r+1)} \\ \overline{B}^{(r+1)} \\ \overline{C}^{(r+1)} \end{bmatrix} = \begin{bmatrix} \overline{A}^{(r)} \#_{\frac{2}{3}} (\overline{B}^{(r)} \# \overline{C}^{(r)}) \\ \overline{B}^{(r)} \#_{\frac{2}{3}} (\overline{A}^{(r)} \# \overline{C}^{(r)}) \\ \overline{C}^{(r)} \#_{\frac{2}{3}} (\overline{A}^{(r)} \# \overline{B}^{(r)}) \end{bmatrix}.$$

We can interpret this iteration as a geometrical construction in the following way. To find e.g. $\overline{A}^{(r+1)}$, the algorithm is:

1. Draw the geodesic joining $\overline{B}^{(r)}$ and $\overline{C}^{(r)}$, and take its midpoint $M^{(r)}$;
2. Draw the geodesic joining $\overline{A}^{(r)}$ and $M^{(r)}$, and take the point lying at $\frac{2}{3}$ of its length: this is $\overline{A}^{(r+1)}$.

If we execute the same algorithm on the Euclidean plane, replacing the word "geodesic" with "straight line segment", it turns out that $\overline{A}^{(1)}$, $\overline{B}^{(1)}$, and $\overline{C}^{(1)}$ coincide in the centroid of the triangle with vertices A, B, C. Thus, unlike the Euclidean counterpart of the Ando–Li–Mathias mean, this process converges in one step on the plane. Roughly speaking, when A, B and C are very close to each other, we can approximate (in some intuitive sense) the geometry on the Riemannian manifold $\mathbb{P}^n$ with the geometry on the Euclidean plane: since this construction to find the centroid of a plane triangle converges faster than the Ando–Li–Mathias one, we can expect that also the convergence speed of the resulting algorithm is faster. This is indeed what results after a more accurate convergence analysis.

Global convergence and properties P1–P11

In order to prove that the iteration (12.4) is convergent (and thus that $\overline{G}(\cdot)$ is well defined), we adapt part of the proof of Theorem 3.2 of [ALM04] (namely, Argument 1).

Theorem 12.1 ([BMP10a])**.** *Let $A_1, \dots A_k$ be positive definite.*

1. *All the sequences $(\overline{A}_i^{(r)})_{r=1}^{\infty}$ converge for $r \to \infty$ to a common limit $\overline{A}$;*
2. *the function $\overline{G}_k(A_1, \dots, A_k)$ satisfies P1–P11.*

Proof. We work by induction on k. For $k = 2$, our mean coincides with the ALM mean, so all the required work has been done in [ALM04]. Let us now suppose that the thesis holds for all $k' \leq k - 1$. We have

$$\overline{A}_i^{(r+1)} \leq \frac{1}{k} \left(\overline{A}_i^{(r)} + (k-1) T_i^{(r)} \right) \leq \frac{1}{k} \sum_{i=1}^{k} \overline{A}_i^{(r)},$$

where the first inequality follows from P10 for the ALM mean $G_k(\cdot)$ (remember that in the special case in which $k - 1$ of the arguments coincide, $G_k(\cdot) = \overline{G}_k(\cdot)$), and the second from P10 for $\overline{G}_{k-1}(\cdot)$. Thus,

$$\sum_{i=1}^{k} \overline{A}_i^{(r+1)} \leq \sum_{i=1}^{k} \overline{A}_i^{(r)} \leq \sum_{i=1}^{k} A_i. \tag{12.5}$$

Therefore, the sequence $(\overline{A}_1^{(r)}, \dots, \overline{A}_k^{(r)})_{r=1}^{\infty}$ is bounded, and there must be a converging subsequence, say, converging to $(\overline{A}_1, \dots, \overline{A}_k)$.

Moreover, for each $p, q \in \{1, \dots, k\}$ we have

$$\begin{aligned} R(\overline{A}_p^{(r+1)}, \overline{A}_q^{(r+1)}) &\le R(\overline{A}_p^{(r)}, \overline{A}_q^{(r)})^{1/k} R(T_p^{(r)}, T_q^{(r)})^{\frac{k-1}{k}} \\ &\le R(\overline{A}_p^{(r)}, \overline{A}_q^{(r)})^{1/k} (R(\overline{A}_q^{(r)}, \overline{A}_p^{(r)})^{\frac{1}{k-1}})^{\frac{k-1}{k}} = R(\overline{A}_p^{(r)}, \overline{A}_q^{(r)})^{2/k}, \end{aligned}$$

where the first inequality follows from P11 in the special case, and the second from P11 in the inductive hypothesis. Passing at the limit of the converging subsequence, one can verify that

$$R(\overline{A}_p, \overline{A}_q) \le R(\overline{A}_p, \overline{A}_q)^{2/k},$$

from which we get $R(\overline{A}_p, \overline{A}_q) \le 1$, that is, $\overline{A}_p = \overline{A}_q$, because of the properties of R, i.e., the limit of the subsequence is in the form $(\overline{A}, \overline{A}, \dots, \overline{A})$. Suppose there is another subsequence converging to $(\overline{B}, \overline{B}, \dots, \overline{B})$; then, by (12.5), we have both $k\overline{A} \le k\overline{B}$ and $k\overline{B} \le k\overline{A}$, that is, $\overline{A} = \overline{B}$. Therefore, the sequence has only one limit point, thus it is convergent. This proves the first part of the theorem.

We now turn to show that P11 holds for our mean $\overline{G}_k(\cdot)$. Consider k-tuples $A_1, \dots, A_k$ and $B_1, \dots, B_k$, and let $\overline{B}_i^{(r)}$ be defined as $\overline{A}_i^{(r)}$ but starting the iteration from the k-tuple (B_i) instead of (A_i). We have for each i

$$\begin{aligned} R(\overline{A}_i^{(r+1)}, \overline{B}_i^{(r+1)}) \le& \\ \le& R(\overline{A}_i^{(r)}, \overline{B}_i^{(r)})^{1/k} R(\overline{G}(\mathcal{Z}_i(\overline{A}_1^{(r)}, \dots, \overline{A}_k^{(r)})), \overline{G}(\mathcal{Z}_i(\overline{B}_1^{(r)}, \dots, \overline{B}_k^{(r)})))^{\frac{k-1}{k}} \\ \le& R(\overline{A}_i^{(r)}, \overline{B}_i^{(r)})^{1/k} \left(\prod_{j \ne i} R(\overline{A}_j^{(r)}, \overline{B}_j^{(r)})^{\frac{1}{k-1}} \right)^{\frac{k-1}{k}} \\ =& \prod_{j=1}^{k} R(\overline{A}_j^{(r)}, \overline{B}_j^{(r)})^{1/k}. \end{aligned}$$

This yields

$$\prod_{i=1}^{k} R(\overline{A}_i^{(r+1)}, \overline{B}_i^{(r+1)}) \le \prod_{i=1}^{k} R(\overline{A}_i^{(r)}, \overline{B}_i^{(r)});$$

chaining together these inequalities for successive values of r and passing to the limit, we get

$$R(G(A_1, \dots, A_k), G(B_1, \dots, B_k))^k \le \prod_{i=1}^{k} R(A_i, B_i),$$

which is P11.

The other properties P1–P4 and P6–P9 (remember that P5 and P10 are consequences of these) are not difficult to prove. All the proofs are quite similar, and can be established by induction, using also the fact that since they hold for the ALM mean, they can be applied to the mean $\overline{G}(\cdot)$ appearing in (12.4) (since we just proved that all possible geometric means take the same value if applied with $k-1$ equal arguments). For the sake of brevity, we provide only the proof for three of these properties.

P1 We need to prove that if the A_i commute then $\overline{G}(A_1, \dots, A_k) = (A_1 \cdots A_k)^{1/k}$. Using the inductive hypothesis, we have $T_i^{(1)} = \prod_{j \ne i} \overline{A}_i^{\frac{1}{k-1}}$. Using the fact that P1 holds for

the ALM mean, we have

$$\overline{A}_i^{(1)} = A_i^{1/k} \left(\prod_{j \neq i} A_j^{\frac{1}{k-1}} \right)^{\frac{k-1}{k}} = \prod_{i=1}^{k} A_i^{1/k},$$

as needed. So, from the second iteration on, we have $\overline{A}_1^{(r)} = \overline{A}_2^{(r)} = \cdots = \overline{A}_k^{(r)} = \prod_{i=1}^k A_i^{1/k}$.

P4 Let $T_i'^{(r)}$ and $\overline{A}_i'^{(r)}$ be defined as $T_i^{(r)}$ and $\overline{A}_i^{(r)}$ but starting from $A_i' \leq A_i$. Using monotonicity in the inductive case and in the ALM mean, we have for each $s \leq 1$ and for each i

$$T_i^{(r+1)} \leq T_i'^{(r+1)}$$

and thus

$$\overline{A}_i^{(r+1)} \leq \overline{A}_i'^{(r+1)}.$$

Passing to the limit for $r \to \infty$, we obtain P4.

P7 Suppose $A_i = \lambda A_i' + (1-\lambda)A_i''$, and let $T_i'^{(r)}$, (resp. $T_i''^{(r)}$) and $\overline{A}_i'^{(r)}$, (resp. $\overline{A}_i''^{(r)}$) be defined as $T_i^{(r)}$ and $\overline{A}_i^{(r)}$ but starting from A_i' (resp. A_i''). Suppose that for some r it holds $\overline{A}_i^{(r)} \geq \lambda \overline{A}_i'^{(r)} + (1-\lambda)\overline{A}_i''^{(r)}$ for all i. Then by joint concavity and monotonicity in the inductive case we have

$$\begin{aligned} T_i^{(r+1)} =& \overline{G}(\mathcal{Z}_i(\overline{A}_1^{(r)}, \ldots, \overline{A}_k^{(r)})) \\ \geq& \overline{G}(\mathcal{Z}_i(\lambda \overline{A}_1'^{(r)} + (1-\lambda)\overline{A}_1''^{(r)}, \ldots, \lambda \overline{A}_k'^{(r)} + (1-\lambda)\overline{A}_k''^{(r)})) \\ \geq& \lambda T_i'^{(r)} + (1-\lambda) T_i''^{(r)}, \end{aligned}$$

and by joint concavity and monotonicity of the Ando–Li–Mathias mean we have

$$\begin{aligned} \overline{A}_i^{(r+1)} =& \overline{A}_i^{(r)} \#_{\frac{k-1}{k}} T_i^{(r)} \\ \geq& \left(\lambda \overline{A}_i'^{(r)} + (1-\lambda)\overline{A}_i''^{(r)} \right) \#_{\frac{k-1}{k}} \left(\lambda T_i'^{(r)} + (1-\lambda) T_i''^{(r)} \right) \\ \geq& \lambda \overline{A}_i'^{(r+1)} + (1-\lambda)\overline{A}_i''^{(r+1)}. \end{aligned}$$

Passing to the limit for $r \to \infty$, we obtain P7.

□

Cubic convergence

In this section, we use the big-O notation in the norm sense, that is, we write $X = Y + O(\varepsilon^h)$ to denote that there are universal positive constants $\varepsilon_0 < 1$ and θ such that for each $0 < \varepsilon < \varepsilon_0$ it holds $\|X - Y\| \leq \theta \varepsilon^h$. The usual arithmetic rules involving this notation hold. In the following, these constants may depend on k, but not on the specific choice of the matrices involved in the formulas.

Theorem 12.2 ([BMP10a]). *Let $0 < \varepsilon < 1$, M and $\overline{A}_i^{(0)} = A_i$, $i = 1, \ldots, k$, be positive definite $n \times n$, and $E_i := M^{-1}A_i - I$. Suppose that $\|E_i\| \leq \varepsilon$ for all $i = 1, \ldots, k$. Then, for the matrices $\overline{A}_i^{(1)}$ defined in* (12.4) *the following hold.*

C1 *We have*

$$M^{-1}\overline{A}_i^{(1)} - I = T_k + O(\varepsilon^3) \tag{12.6}$$

where

$$T_k := \frac{1}{k}\sum_{j=1}^{k} E_j - \frac{1}{4k^2}\sum_{i,j=1}^{k}(E_i - E_j)^2.$$

C2 *There are positive constants θ, σ and $\overline{\varepsilon} < 1$ (all of which may depend on k) such that for all $\varepsilon \le \overline{\varepsilon}$ it holds*

$$\left\| M_1^{-1}\overline{A}_i^{(1)} - I \right\| \le \theta\varepsilon^3$$

for a suitable matrix M_1 satisfying $\left\|M^{-1}M_1 - I\right\| \le \sigma\varepsilon$.

C3 *The iteration* (12.4) *converges at least cubically.*

C4 *We have*

$$M_1^{-1}\overline{G}(A_1, \dots, A_k) - I = O(\varepsilon^3). \tag{12.7}$$

Proof. Let us first find a local expansion of a generic point on the geodesic $A \#_t B$: let $M^{-1}A = I + F_1$ and $M^{-1}B = I + F_2$ with $\|F_1\| \le \delta$, $\|F_2\| \le \delta$, $0 < \delta < 1$. Then we have

$$\begin{aligned}
M^{-1}(A \#_t B) =& M^{-1}A(A^{-1}B)^t = (I + F_1)\left((I + F_1)^{-1}(I + F_2)\right)^t \\
=& (I + F_1)\left((I - F_1 + F_1^2 + O(\delta^3))(I + F_2)\right)^t \\
=& (I + F_1)\left(I + F_2 - F_1 - F_1F_2 + F_1^2 + O(\delta^3)\right)^t \\
=& (I + F_1)\Big(I + t(F_2 - F_1 - F_1F_2 + F_1^2) \\
&+ \frac{t(t-1)}{2}(F_2 - F_1)^2 + O(\delta^3)\Big) \\
=& I + (1-t)F_1 + tF_2 + \frac{t(t-1)}{2}(F_2 - F_1)^2 + O(\delta^3),
\end{aligned} \tag{12.8}$$

where we made use of the matrix series expansion $(I + X)^t = I + tX + \frac{t(t-1)}{2}X^2 + O(X^3)$.

We prove the theorem by induction on k in the following way. Let Ci_k denote the assertion Ci of the theorem (for $i = 1, \dots 4$) for a given value of k. We show that

1. $\mathrm{C1}_2$ holds;
2. $\mathrm{C1}_k \Longrightarrow \mathrm{C2}_k$;
3. $\mathrm{C2}_k \Longrightarrow \mathrm{C3}_k, \mathrm{C4}_k$;
4. $\mathrm{C4}_k \Longrightarrow \mathrm{C1}_{k+1}$.

It is clear that these claims imply that the results C1—C4 hold for all $k \ge 2$ by induction; we now turn to prove them one by one.

1. This is simply equation (12.8) for $t = \frac{1}{2}$.
2. It is obvious that $T_k = O(\varepsilon)$; thus, choosing $M_1 := M(I + T_k)$ one has

$$\overline{A}_i^{(1)} = M(I + T_k + O(\varepsilon^3)) = M_1(I + (I + T_k)^{-1}O(\varepsilon^3)) = M_1(I + O(\varepsilon^3)). \tag{12.9}$$

Using explicit constants in the big-O estimates, we get

$$\left\|M_1^{-1}\overline{A}_i^{(1)} - I\right\| \leq \theta\varepsilon^3, \quad \left\|M^{-1}M_1 - I\right\| \leq \sigma\varepsilon$$

for suitable constants θ and σ.

3. Suppose ε is small enough to have $\theta\varepsilon^3 \leq \varepsilon$. We apply C2 with initial matrices $\overline{A}_i^{(1)}$, with $\varepsilon_1 = \theta\varepsilon^3$ in lieu of ε and M_1 in lieu of M, getting

$$\left\|M_2^{-1}\overline{A}_i^{(2)} - I\right\| \leq \theta\varepsilon_1^3, \quad \left\|M_1^{-1}M_2 - I\right\| \leq \sigma\varepsilon_1.$$

Repeating again for all the steps of our iterative process, we get for all $s = 1, 2, \ldots$

$$\left\|M_s^{-1}\overline{A}_i^{(s)} - I\right\| \leq \theta\varepsilon_{s-1}^3 = \varepsilon_s, \quad \left\|M_s^{-1}M_{s+1} - I\right\| \leq \sigma\varepsilon_s \tag{12.10}$$

with $\varepsilon_{s+1} := \theta\varepsilon_s^3$ and $M_0 := M$.

For simplicity's sake, we introduce the notation

$$d(X, Y) := \left\|X^{-1}Y - I\right\|$$

for any two $n \times n$ symmetric positive definite matrices X and Y. It is useful to notice that $\|X - Y\| \leq \|X\| \left\|X^{-1}Y - I\right\| \leq \|X\|\, d(X, Y)$ and

$$\begin{aligned} d(X, Z) = &\left\|(X^{-1}Y - I)(Y^{-1}Z - I) + X^{-1}Y - I + Y^{-1}Z - I\right\| \\ \leq &d(X, Y)d(Y, Z) + d(X, Y) + d(Y, Z). \end{aligned} \tag{12.11}$$

With this notation, we can restate (12.10) as

$$d(M_s, \overline{A}_i^{(s)}) \leq \varepsilon_s, \quad d(M_s, M_{s+1}) \leq \sigma\varepsilon_s.$$

We prove by induction that, for ε smaller than a fixed constant, it holds

$$d(M_s, M_{s+t}) \leq \left(2 - \frac{1}{2^t}\right)\sigma\varepsilon_s. \tag{12.12}$$

First of all, it holds for all $t \geq 1$

$$\varepsilon_{s+t} = \theta^{\frac{3^t-1}{2}}\varepsilon^{3^t},$$

which, for ε smaller than $\min(\frac{1}{8}, \theta^{-1})$, implies $\frac{\varepsilon_{s+t}}{\varepsilon_s} \leq \varepsilon_s^{\frac{3^t-1}{2}} \leq \varepsilon_s^t \leq \frac{1}{2^{t+2}}$.

Now, using (12.11), and supposing additionally $\varepsilon \leq \sigma^{-1}$, we have

$$\begin{aligned} d(M_s, M_{s+t+1}) \leq &d(M_s, M_{s+t})d(M_{s+t}, M_{s+t+1}) \\ &+ d(M_s, M_{s+t}) + d(M_{s+t}, M_{s+t+1}) \\ \leq &\left(2 - \frac{1}{2^t}\right)\sigma\varepsilon_s + \sigma\varepsilon_s\left(\sigma\varepsilon_{s+t} + \frac{\varepsilon_{s+t}}{\varepsilon_s}\right) \\ \leq &\left(2 - \frac{1}{2^t}\right)\sigma\varepsilon_s + \sigma\varepsilon_s\left(2\frac{\varepsilon_{s+t}}{\varepsilon_s}\right) \\ \leq &\left(2 - \frac{1}{2^t}\right)\sigma\varepsilon_s + \sigma\varepsilon_s\frac{1}{2^{t+1}} = \left(2 - \frac{1}{2^{t+1}}\right)\sigma\varepsilon_s \end{aligned}$$

Thus, we have for each t

$$\|M_t - M\| \le \|M\| \left\|M^{-1}M_t - I\right\| \le 2\sigma \|M\| \varepsilon,$$

which implies $\|M_t\| \le 2\|M\|$ for all t. By a similar argument,

$$\|M_{s+t} - M_s\| \le \|M_s\| d(M_{s+t}, M_s) \le 2\sigma \|M\| \varepsilon_s, \tag{12.13}$$

Due to the bounds already imposed on ε, the sequence ε_s tends monotonically to zero with cubic convergence rate; thus (M_t) is a Cauchy sequence and therefore converges. In the following, let M^* be its limit. The convergence rate is cubic, since passing to the limit (12.13) we get

$$\|M^* - M_s\| \le 2\sigma \|M\| \varepsilon_s.$$

Now, using the other relation in (12.10), we get

$$\begin{aligned}\left\|\overline{A}_i^{(s)} - M^*\right\| \le& \left\|\overline{A}_i^{(s)} - M^s\right\| + \|M^* - M_s\| \\ \le& 2\|M\| d(M_s, \overline{A}_i^{(s)}) + 2\sigma \|M\| \varepsilon_s \\ \le& (2\sigma + 2)\|M\| \varepsilon_s,\end{aligned}$$

that is, $\overline{A}_i^{(s)}$ converges with cubic convergence rate to M^*. Thus C3 is proved. By (12.11), (12.12), and (12.10), we have

$$\begin{aligned}d(M_1, \overline{A}_i^{(t)}) \le& d(M_1, M_t) d(M_t, \overline{A}_i^{(t)}) + d(M_1, M_t) + d(M_t, \overline{A}_i^{(t)}) \\ \le& 2\sigma\varepsilon_1\varepsilon_t + 2\sigma\varepsilon_1 + \varepsilon_t \le (4\sigma + 1)\varepsilon_1 = O(\varepsilon^3),\end{aligned}$$

which is C4.

4. Using $C4_k$ and (12.8) with $F_1 = E_{k+1}$, $F_2 = M^{-1}\overline{G}(A_1, \dots, A_k) = T_k + O(\varepsilon^3)$, $\delta = 2k\varepsilon$, we have

$$\begin{aligned}M^{-1}\overline{A}_{k+1}^{(1)} =& M^{-1}\left(A_{k+1} \#_{\frac{k}{k+1}} \overline{G}(A_1, \dots, A_k)\right) \\ =& I + \frac{1}{k+1}E_{k+1} + \frac{k}{k+1}T_k \\ & - \frac{k}{2(k+1)^2}\left(E_{k+1} - \frac{1}{k}\sum_{i=1}^{k} E_i\right)^2 + O(\varepsilon^3).\end{aligned} \tag{12.14}$$

Observe that

$$T_k = \frac{1}{k}S_k + \frac{P_k - (k-1)Q_k}{2k^2}$$

where $S_k = \sum_{i=1}^k E_i$, $Q_k = \sum_{i=1}^k E_i^2$, $P_k = \sum_{i,j=1,\ i\neq j}^k E_iE_j$. Since $S_k^2 = P_k + Q_k$ and $S_{k+1} = S_k + E_{k+1}$, $Q_{k+1} = Q_k + E_{k+1}^2$, $P_{k+1} = P_k + E_{k+1}S_k + S_kE_{k+1}$, from (12.14) one finds that

$$\begin{aligned}M^{-1}\overline{A}_{k+1}^{(1)} =& I + \frac{1}{k+1}S_{k+1} - \frac{k}{2(k+1)^2}Q_{k+1} + \frac{1}{2(k+1)^2}P_{k+1} + O(\varepsilon^3) \\ =& I + T_{k+1} + O(\varepsilon^3).\end{aligned}$$

Since the expression we found is symmetric with respect to the E_i, it follows that $\overline{A}_j^{(1)}$ has the same expansion for any j.

□

Observe that Theorems 12.1 and 12.2 imply that the iteration (12.4) is globally convergent with order of convergence at least 3.

It is worth to point out that, in the case where the matrices A_i, $i = 1, \ldots, A_k$, commute each other, the iteration (12.4) converges in just one step, i.e., $\overline{A}_i^{(1)} = \overline{A}$ for any i. In the noncommutative general case, one has $\det(\overline{A}_i^{(s)}) = \det(\overline{A})$ for any i and for any $s \geq 1$, i.e., the determinant converges in one single step to the determinant of the matrix mean.

Our mean is different from the ALM mean, as we show with some numerical experiments in Section 12.5.

12.4 A new class of matrix geometric means

In this section we introduce a new class of matrix means depending on a set of parameters $s_1, \ldots, s_{k-1}$ and show that the ALM mean and our mean are two specific instances of this class.

For the sake of simplicity, we describe this generalization in the case of $k = 3$ matrices A, B, C. The case $k > 3$ is outlined. Here, the distance between two matrices is defined in (12.1).

For $k = 3$, the algorithm presented in Section 12.3 replaces the triple A, B, C with A', B', C' where A' is chosen in the geodesic connecting A with the midpoint of the geodesic connecting B and C, at distance $\frac{2}{3}$ from A, and similarly is made for B' and C'. In our generalization we use two parameters $s, t \in [0, 1]$. We consider the point $P_t = B \#_t C$ in the geodesic connecting B to C at distance t from B. Then we consider the geodesic connecting A to P_t and define A' the matrix on this geodesic at distance s from A. That is, we set $A' = A \#_s (B \#_t C)$. Similarly we do with B and C. This transformation is recursively repeated so that the matrix sequences $A^{(r)}$, $B^{(r)}$, $C^{(r)}$ are generated by means of

$$\begin{aligned} A^{(r+1)} =& A^{(r)} \#_s (B^{(r)} \#_t C^{(r)}), \\ B^{(r+1)} =& B^{(r)} \#_s (C^{(r)} \#_t A^{(r)}), \qquad\qquad r = 0, 1, \ldots, \\ C^{(r+1)} =& C^{(r)} \#_s (A^{(r)} \#_t B^{(r)}), \end{aligned} \tag{12.15}$$

starting with $A^{(0)} = A$, $B^{(0)} = B$, $C^{(0)} = C$.

By following the same argument of Section 12.3 it can be shown that the three sequences have a common limit $G_{s,t}$ for any $s, t \in [0, 1]$. Moreover, for $s = 1$, $t = \frac{1}{2}$ one obtains the ALM mean, i.e., $G = G_{1,\frac{1}{2}}$, while for $s = \frac{2}{3}$, $t = \frac{1}{2}$ the limit coincides with our mean, i.e., $\overline{G} = G_{\frac{2}{3},\frac{1}{2}}$. Moreover, it is possible to prove that for any $s, t \in [0, 1]$ the limit satisfies the conditions P1–P11 so that it can be considered a good geometric mean.

Concerning the convergence speed of the sequence generated by (12.15) we may perform a more accurate analysis. Assume that $A = M(I + E_1)$, $B = M(I + E_2)$, $C = M(I + E_3)$, where $\|E_i\| \leq \varepsilon < 1$, $i = 1, 2, 3$. Then, applying (12.8) in (12.15) yields

$$\begin{aligned} A' \doteq& M(I + (1-s)E_1 + s(1-t)E_2 + stE_3 + \frac{st(t-1)}{2}H_2^2 + \frac{s(s-1)}{2}(H_1 + tH_2)^2) \\ B' \doteq& M(I + (1-s)E_2 + s(1-t)E_3 + stE_1 + \frac{st(t-1)}{2}H_3^2 + \frac{s(s-1)}{2}(H_2 + tH_3)^2) \\ C' \doteq& M(I + (1-s)E_3 + s(1-t)E_1 + stE_2 + \frac{st(t-1)}{2}H_1^2 + \frac{s(s-1)}{2}(H_3 + tH_1)^2) \end{aligned}$$

where $\doteq$ denotes equality up to $O(\varepsilon^3)$ terms, with $H_1 = E_1 - E_2$, $H_2 = E_2 - E_3$, $H_3 = E_3 - E_1$. Whence we have $A' = M(I + E_1')$, $B' = M(I + E_2')$, $C' = M(I + E_3')$, with

$$\begin{bmatrix} E_1' \\ E_2' \\ E_3' \end{bmatrix} \doteq C(s,t) \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} + \frac{st(t-1)}{2} \begin{bmatrix} H_2^2 \\ H_3^2 \\ H_1^2 \end{bmatrix} + \frac{s(s-1)}{2} \begin{bmatrix} (H_1 - tH_2)^2 \\ (H_2 - tH_3)^2 \\ (H_3 - tH_1)^2 \end{bmatrix}$$

where

$$C(s,t) = \begin{bmatrix} (1-s)I & s(1-t)I & stI \\ stI & (1-s)I & s(1-t)I \\ s(1-t)I & stI & (1-s)I \end{bmatrix}.$$

Observe that the block circulant matrix $C(s,t)$ has eigenvalues $\lambda_1 = 1$, $\lambda_2 = (1 - \frac{3}{2}s) + \boldsymbol{i}\frac{\sqrt{3}}{2}s(2t-1)$, and $\lambda_3 = \overline{\lambda}_2$, with multiplicity n, where $\boldsymbol{i}^2 = -1$. Moreover, the pair $(s,t) = (\frac{2}{3}, \frac{1}{2})$ is the only one which yields $\lambda_2 = \lambda_3 = 0$. In fact $(\frac{2}{3}, \frac{1}{2})$ is the only pair which provides superlinear convergence. For the ALM mean, where $t = \frac{1}{2}$ and $s = 1$ it holds $|\lambda_2| = |\lambda_3| = \frac{1}{2}$ which is the rate of convergence of the ALM iteration [ALM04].

In the case of $k > 3$ matrices, given the $(k-1)$-tuple $(s_1, s_2, \ldots, s_{k-1})$ we may recursively define $G_{s_1,\ldots,s_{k-1}}(A_1, \ldots, A_k)$ as the common limit of the sequences generated by

$$A_i^{(r+1)} = A_i^{(r)} \#_{s_1} G_{s_2,\ldots,s_{k-1}}(\mathcal{Z}_i(A_1^{(r)}, \ldots, A_k^{(r)})), \qquad i = 1, \ldots, k.$$

Observe that with $(s_1, \ldots, s_{k-1}) = (\frac{1}{2}, 1, 1, \ldots, 1)$ one obtains the ALM mean, while with $(s_1, \ldots, s_{k-1}) = (\frac{k-1}{k}, \frac{k-2}{k-1}, \ldots, \frac{1}{2})$ one obtains the new mean introduced in Section 12.3.

12.5 Numerical experiments

We have implemented the two iterations converging to the ALM mean and to the newly defined geometric mean in Matlab, and run some numerical experiments on a quad-Xeon 2.8Ghz computer. To compute matrix square roots we used Matlab's built-in `sqrtm` function, while for p-th roots with $p > 2$ we used the `rootm` function in Nicholas Higham's Matrix Computation Toolbox [Higa]. To counter the loss of symmetry due to the accumulation of computational errors, we chose to discard the imaginary part of the computed roots.

The experiments have been performed on the same data set as the paper [Moa06]. It consists of five sets each composed of four to six 6×6 positive definite matrices, corresponding to physical data from elasticity experiments conducted by Hearmon [Hea52]. The matrices are composed of smaller diagonal blocks of sizes 1×1 up to 4×4, depending on the symmetries of the involved materials. Two to three significant digits are reported for each experiments.

We have computed both the ALM mean and the newly defined mean of these sets; as a stopping criterion for each computed mean, we chose

$$\max_i \left| A_i^{(r+1)} - A_i^{(r)} \right| < \varepsilon,$$

where $|X| := \max_{i,j} |X_{ij}|$, with $\varepsilon = 10^{-10}$. The CPU times, in seconds, are reported in Table 12.1. For four matrices, the speed gain is a factor of 20, and it increases even more for more than four matrices.

We then focused on Hearmon's second data set (ammonium dihydrogen phosphate), composed of four matrices. In Table 12.2, we reported the number of outer ($k = 4$) iterations

Data set (number of matrices)	ALM mean	New mean
$NaClO_3$ (5)	$2.3 \times 10^{+2}$	$1.3 \times 10^{+0}$
Ammonium dihydrogen phosphate (4)	$9.9 \times 10^{+0}$	3.9×10^{-1}
Potassium dihydrogen phosphate (4)	$9.7 \times 10^{+0}$	3.8×10^{-1}
Quartz (6)	$6.7 \times 10^{+3}$	$3.0 \times 10^{+1}$
Rochelle salt (4)	$1.0 \times 10^{+1}$	5.3×10^{-1}

Table 12.1: CPU times in seconds for the Hearmon elasticity data

	ALM mean	New mean
Outer iterations	23	3
Avg. inner iterations	18.3	2
Matrix square roots (`sqrtm`)	5,052	72
Matrix p-th roots (`rootm`)	0	84

Table 12.2: Number of inner and outer iterations needed, and number of matrix roots needed

Operation	Result
$\left\lvert G(A^4, I, I, I) - A\right\rvert$	3.6×10^{-13}
$\left\lvert \overline{G}(A^4, I, I, I) - A\right\rvert$	1.8×10^{-14}

Table 12.3: Accuracy of two geometric mean computations

needed and the average number of iterations needed to reach convergence in the inner ($k = 3$) iterations (remember that the computation of a mean of four matrices requires the computation of three means of three matrices at each of its steps). Moreover, we measured the number of square and p-th roots needed by the two algorithms, since they are the most expensive operation in the algorithm. From the results, it is evident that the speed gain in the new mean is due not only to the reduction of the number of outer iterations, but also of the number of inner iterations needed to get convergence at each step of the inner mean calculations. When the number of involved matrices becomes larger, these speedups add up at each level.

Hearmon's elasticity data are not suitable to measure the accuracy of the algorithm, since the results to be obtained are not known. To measure the accuracy of the computed results, we computed instead $\left\lvert G(A^4, I, I, I) - A\right\rvert$, which should yield zero in exact arithmetic (due to P1), and its analogue with the new mean. We chose A to be the first matrix in Hearmon's second data set. Moreover, in order to obtain results closer to machine precision, in this experiment we changed the stopping criterion choosing $\varepsilon = 10^{-13}$.

The results reported in Table 12.3 are well within the errors permitted by the stopping criterion, and show that both algorithms can reach a satisfying precision.

The following examples provide an experimental proof that our mean is different from the ALM mean.

Consider the following matrices

$$A = \begin{bmatrix} a & b \\ b & a \end{bmatrix} \qquad B = \begin{bmatrix} a & -b \\ -b & a \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 \\ 0 & c \end{bmatrix}.$$

Observe that the triple (A, B, C) is transformed into (B, A, C) under the map $X \to S^{-1}XS$, for $S = \operatorname{diag}(1, -1)$. In this way, any matrix mean $G(A, B, C)$ satisfying condition P3 is such that $G = S^{-1}GS$, that is, the off-diagonal entries of G are zero. Whence, G must be diagonal. With $a = 2, b = 1, c = 24$, for the ALM mean G and our mean $\overline{G}$ one finds that

$$\overline{G} = \begin{bmatrix} 1.487443626 & 0 \\ 0 & 4.033766318 \end{bmatrix}, \qquad G = \begin{bmatrix} 1.485347837 & 0 \\ 0 & 4.039457861 \end{bmatrix},$$

where we reported the first 10 digits. Observe that the determinant of both the matrices is 6, that is, the geometric mean of $\det A, \det B, \det C$, moreover, $\rho(\overline{G}) < \rho(G)$.

For the matrices

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -2 \\ 0 & -2 & 2 \end{bmatrix}, \qquad B = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 2 \\ 0 & 2 & 2 \end{bmatrix},$$
$$C = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 10 & 0 \\ 1 & 0 & 50 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 10 & 0 \\ -1 & 0 & 50 \end{bmatrix},$$

one has

$$\overline{G} = \begin{bmatrix} 1.3481 & 0 & -0.3016 \\ 0 & 3.8452 & 0 \\ -0.3016 & 0 & 6.1068 \end{bmatrix}, \qquad G = \begin{bmatrix} 1.3472 & 0 & -0.3106 \\ 0 & 3.8796 & 0 \\ -0.3106 & 0 & 6.0611 \end{bmatrix}.$$

Their eigenvalues are $(6.1258, 3.8452, 1.3290)$, and $(6.0815, 3.8796, 1.3268)$, respectively. Observe that, unlike in the previous example, it holds $\rho(\overline{G}) > \rho(G)$.

12.6 Conclusions and research lines

In this chapter, we have described a new matrix geometric mean which is constructed with an iteration similar to the one proposed by Ando, Li and Mathias, but it converges cubically instead of linearly. Therefore, its computation requires far less CPU time than the original ALM mean. However, the time required still grows as $O(k!)$ with the number k of matrices to average. This is an important drawback which prevents from computing matrix means of large numbers of matrices. As we showed in the examples, even averaging six matrices takes 30 seconds on a modern machine. The problem of reducing this complexity is addressed in the next chapter.

Chapter 13

Constructing other matrix geometric means

13.1 Introduction

In the last few years, several papers have been devoted to defining a proper way to generalize the concept of geometric mean to $n \geq 3$ Hermitian, positive definite $m \times m$ matrices. Chapter 12 mention the desirable properties listed by Ando, Li and Mathias [ALM04]; however, these properties do not uniquely define a multivariate matrix geometric mean; thus several different definitions appeared in literature.

We have already introduced in Chapter 12 the Ando–Li–Mathias mean [ALM04] and a new mean first described in [BMP10a]. Both are based on iterations reducing a mean of n matrices to one of $n-1$ matrices and satisfy all the desired properties, but their computation requires a number of operations which grows as $O(n!)$ with the number of involved matrices; therefore, they become quickly impractical even for a moderate number of matrices. Pálfia [Pál09] proposed a mean based on a similar process involving only means of 2 matrices, and thus much cheaper to compute, but lacking property P3 (permutation invariance) from the ALM list. Lim [Lim08] proposed a family of matrix geometric means that are based on an iteration requiring at each step the computation of a mean of $m \geq n$ matrices. Since the computational complexity for all known means greatly increases with n, the resulting family is useful as an example but highly impractical for numerical computations.

At the same time, Moakher [Moa05, Moa06] and Bhatia and Holbrook [BH06b, Bha07] proposed a completely different definition, which we call the *Riemannian centroid* of $A_1, A_2, \ldots, A_n$. The Riemannian centroid $G^R(A_1, A_2, \ldots, A_n)$ is defined as the minimizer of a sum of squared distances,

$$G^R(A_1, A_2, \ldots, A_n) = \arg\min_X \sum_{i=1}^{n} \delta^2(A_i, X), \tag{13.1}$$

where δ is the geodesic distance induced by a natural Riemannian metric on the space of symmetric positive definite matrices. The same X is the unique solution of the equation

$$\sum_{i=1}^{n} \log(A_i^{-1} X) = 0, \tag{13.2}$$

involving the matrix logarithm function. While most of the ALM properties are easy to prove, it is still an open problem whether it satisfies P4 (monotonicity). The computational

experiments performed up to now gave no counterexamples, but the monotonicity of the Riemannian centroid is still a conjecture [BH06b], up to our knowledge.

Moreover, while the other means had constructive definitions, it is not apparent how to compute the solution to either (13.1) or (13.2). Two methods have been proposed, one based on a fixed-point iteration [Moa06] and one on the Newton methods for manifolds [Pál09, Moa06]. Although both seem to work well on "tame" examples, their computational results show a fast degradation of the convergence behavior as the number of matrices and their dimension increase. It is unclear whether on more complicated examples there is convergence in the first place; unlike the other means, the convergence of these iteration processes has not been proved, as far as we know.

Notation

We recall that throughout this chapter we use the positive-definite ordering on matrices, and that we denote by $\mathbb{P}^m$ the space of Hermitian positive-definite $m \times m$ matrices. We say that $\underline{A} = (A_i)_{i=1}^n \in (\mathbb{P}^m)^n$ is a *scalar* n-tuple of matrices if $A_1 = A_2 = \cdots = A_n$. We use the convention that both $Q(\underline{A})$ and $Q(A_1, \ldots, A_n)$ denote the application of the map $Q : (\mathbb{P}^m)^n \to \mathbb{P}^m$ to the n-tuple $\underline{A}$.

13.2 Many examples of (quasi-)means

The matrix geometric mean for $n = 2$

For $n = 2$, the ALM properties uniquely define a matrix geometric mean which can be expressed explicitly as

$$A \# B := A(A^{-1}B)^{1/2}. \tag{13.3}$$

This is a particular case of the more general map

$$A \#_t B := A(A^{-1}B)^t, \quad t \in \mathbb{R}, \tag{13.4}$$

which has a geometrical interpretation as the parametrization of the geodesic joining A and B for a certain Riemannian geometry on $\mathbb{P}^m$ [Bha07].

The ALM and BMP means

Ando, Li and Mathias [ALM04] recursively define a matrix geometric mean G_n^{ALM} of n matrices in this way. The mean G_2^{ALM} of two matrices coincides with (13.3); for $n \geq 3$, suppose the mean of $n-1$ matrices G_{n-1}^{ALM} is already defined. Given $A_1, \ldots, A_n$, compute for each $j = 1, 2, \ldots$

$$A_i^{(j+1)} := G_{n-1}^{ALM}(A_1^{(j)}, A_2^{(j)}, \ldots A_{i-1}^{(j)}, A_{i+1}^{(j)}, \ldots A_n^{(j)}) \quad i = 1, \ldots, n, \tag{13.5}$$

where $A_i^{(0)} := A_i$, $i = 1, \ldots n$. The sequences $(A_i^{(j)})_{j=1}^{\infty}$ converge to a common (not depending on i) matrix, and this matrix is a geometric mean of $A_1^{(0)}, \ldots, A_n^{(0)}$.

The mean proposed by Bini, Meini and Poloni [BMP10a] is defined in the same way, but with (13.5) replaced by

$$A_i^{(j+1)} := G_{n-1}^{BMP}(A_1^{(j)}, A_2^{(j)}, \ldots A_{i-1}^{(j)}, A_{i+1}^{(j)}, \ldots A_n^{(j)}) \#_{1/n} A_i \quad i = 1, \ldots, n. \tag{13.6}$$

Though both maps satisfy the ALM properties, matrices A, B, C exist for which $G^{ALM}(A, B, C) \neq G^{BMP}(A, B, C)$.

While the former iteration converges linearly, the latter converges cubically, and thus allows one to compute a matrix geometric mean with a lower number of iterations. In fact, if we call p_k the average number of iterations that the process giving a mean of k matrices takes to converge (which may vary significantly depending on the starting matrices), the total computational cost of the ALM and BMP means can be expressed as $O(n!p_3p_4\dots p_nm^3)$. The only difference between the two complexity bounds lies in the expected magnitude of the values p_k. The presence of a factorial and of a linear number of factors p_k is undesirable, since it means that the problem scales very badly with n. In fact, already with $n = 7, 8$ and moderate values of m, a large CPU time is generally needed to compute a matrix geometric mean [BMP10a].

The Pálfia mean

Pálfia [Pál09] proposed to consider the following iteration. Let again $A_i^{(0)} := A_i$, $i = 1, \dots, n$. Let us define

$$A_i^{(k+1)} := A_i^{(k)} \# A_{i+1}^{(k)}, \quad i = 1, \dots, n, \tag{13.7}$$

where the indices are taken modulo n, i.e., $A_{n+1}^{(k)} = A_1^{(k)}$ for all k. We point out that the definition in the original paper [Pál09] is slightly different, as it considers several possible orderings of the input matrices, but the means defined there can be put in the form (13.7) up to a permutation of the starting matrices $A_1, \dots, A_n$.

As for the previous means, it can be proved that the iteration (13.7) converges to a scalar n-tuple; we call the common limit of all components $G^P(A_1, \dots, A_n)$. As we noted above, this function does not satisfy P3 (permutation invariance), and thus it is not a geometric mean in the ALM sense.

Other composite means

Apart from the Riemannian centroid, all the other definitions follow the same pattern:

- build new functions of n matrices by taking nested compositions of the existing means—preferably using only means of less than n matrices;
- take the common limit of a set of n functions defined as in the above step.

The possibilities for defining new iterations following this pattern are endless. Ando, Li, Mathias, and Bini, Meini, Poloni chose to use in the first step composite functions using computationally expensive means of $n - 1$ matrices; this led to poor convergence results. Pálfia chose instead to use more economical means of two variables as starting points; this led to better convergence (no $O(n!)$), but to a function which is not symmetric with respect to permutations of its entries (P3, permutation invariance).

As we see in the following, the property P3 is crucial: all the other ones are easily proved for a mean defined as composition/limit of existing means.

A natural question to ask is whether we can build a matrix geometric mean of n matrices as the composition of matrix means of less matrices, without the need of a limit process. Two such unsuccessful attempts are reported in the paper by Ando, Li and Mathias [ALM04], as examples of the fact that it is not easy to define a matrix satisfying P1–P10. The first is

$$G^{4rec}(A, B, C, D) := (A \# B) \#(C \# D). \tag{13.8}$$

Unfortunately, there are matrices such that $(A \# B) \#(C \# D) \neq (A \# C) \#(B \# D)$, so P3 fails. A second attempt is

$$G^{rec}(A, B, C) := (A^{4/3} \# B^{4/3}) \# C^{2/3}, \tag{13.9}$$

where the exponents are chosen so that P1 (consistency with scalars) is satisfied. Again, this function is not symmetric in its arguments, and thus fails to satisfy P3.

A second natural question is whether an iterative scheme such as the ones for G^{ALM}, G^{BMP} and G^{P} can yield P3 without having a $O(n!)$ computational cost. For example, if we could build a scheme similar to the ALM and BMP ones, but using only means of $\frac{n}{2}$ matrices in the recursion, then the $O(n!)$ growth would disappear.

In this chapter, we analyze in more detail the process of "assembling" new matrix means from the existing ones, and show which new means can be found, and what cannot be done because of group-theoretical obstructions related to the symmetry properties of the composed functions. By means of a group-theoretical analysis, we show that for $n = 4$ a new matrix mean exists which is simpler to compute than the existing ones; numerical experiments show that the new definition leads to a significant computational advantage. Moreover, we show that for $n > 4$ the existing strategies of composing matrix means and taking limits cannot provide a mean which is computationally simpler than the existing ones.

13.3 Quasi-means and notation

Let us introduce the following variants to some of the Ando–Li–Mathias properties.

P1" Weak consistency with scalars. There are $\alpha, \beta, \gamma \in \mathbb{R}$ such that if A, B, C commute, then $G(A, B, C) = A^\alpha B^\beta C^\gamma$.

P2" Weak homogeneity. There are $\alpha, \beta, \gamma \in \mathbb{R}$ such that for each $r, s, t > 0$, $G(rA, sB, tC) = r^\alpha s^\beta t^\gamma G(A, B, C)$. Notice that if P1" holds as well, these must be the same α, β, γ (proof: substitute scalar values in P1").

P9' Weak determinant identity. For all $d > 0$, if $\det A = \det B = \det C = d$, then $\det G(A, B, C) = d$.

We call a *quasi-mean* a function $Q : (\mathbb{P}^m)^n \to (\mathbb{P}^m)$ that satisfies P1",P2", P4, P6, P7, P8, P9'. This models expressions which are built starting from basic matrix means but are not symmetric, e.g., $A \# G(B, C, D \# E)$, (13.8), and (13.9).

Theorem 13.1 ([Pol09])**.** *If a quasi-mean Q satisfies P3 (permutation invariance), then it is a geometric mean.*

Proof. From P2" and P3, it follows that $\alpha = \beta = \gamma$. From P9', it follows that if $\det A = \det B = \det C = 1$,

$$2^m = \det Q(2A, 2B, 2C) = \det \left(2^{\alpha+\beta+\gamma} Q(A, B, C)\right) = 2^{m(\alpha+\beta+\gamma)},$$

thus $\alpha + \beta + \gamma = 1$. The two relations combined together yield $\alpha = \beta = \gamma = \frac{1}{3}$. Finally, it is proved in Ando, Li and Mathias [ALM04] that P5 and P10 are implied by the other eight properties P1–P4 and P6–P9. □

For two quasi-means Q and R of n matrices, we write $Q = R$ if $Q(\underline{A}) = R(\underline{A})$ for each n-tuple $\underline{A} \in \mathbb{P}^m$

Group theory notation

The notation $H \leq G$ ($H < G$) means that H is a subgroup (proper subgroup) of G. Let us denote by $\mathfrak{S}_n$ the symmetric group on n elements, i.e., the group of all permutations of the set $\{1, 2, \ldots, n\}$. As usual, the symbol $(a_1 a_2 a_3 \ldots a_k)$ stands for the permutation ("cycle") that maps $a_1 \mapsto a_2$, $a_2 \mapsto a_3$, $\ldots a_{k-1} \mapsto a_k$, $a_k \mapsto a_1$ and leaves the other elements of $\{1, 2, \ldots n\}$ unchanged. Different symbols in the above form can be chained to denote the group operation of function composition; for instance, $\sigma = (13)(24)$ is the permutation $(1, 2, 3, 4) \mapsto (3, 4, 1, 2)$. We denote by $\mathfrak{A}_n$ the alternating group on n elements, i.e., the only subgroup of index 2 of $\mathfrak{S}_n$, and by $\mathfrak{D}_n$ the dihedral group over n elements, with cardinality $2n$. The latter is identified with the subgroup of $\mathfrak{S}_n$ generated by the rotation $(1, 2, \ldots, n)$ and the mirror symmetry $(2, n)(3, n-1)\cdots(\frac{n}{2}, \frac{n}{2}+2)$ (for even values of n) or $(2, n)(3, n-1)\cdots(\frac{n+1}{2}, \frac{n+3}{2})$ (for odd values of n).

Let now $H \leq \mathfrak{S}_n$, and let $\{\sigma_1, \ldots, \sigma_r\} \subset \mathfrak{S}_n$ be a transversal for the right cosets $H\sigma$, i.e., a set of maximal cardinality $r = n!/|H|$ such that $\sigma_j \sigma_i^{-1} \notin H$ for all $i \neq j$. The group $\mathfrak{S}_n$ acts by permutation over the cosets $(H\sigma_1, \ldots, H\sigma_r)$, i.e., for each σ there is a permutation $\tau = \rho_H(\sigma)$ such that

$$(H\sigma_1\sigma, \ldots, H\sigma_r\sigma) = (H\sigma_{\tau(1)}, \ldots, H\sigma_{\tau(r)}).$$

It is easy to check that in this case $\rho_H : \mathfrak{S}_n \to \mathfrak{S}_r$ must be a group homomorphism. Notice that if H is a normal subgroup of $\mathfrak{S}_n$, then the action of $\mathfrak{S}_n$ over the coset space is represented by the quotient group $\mathfrak{S}_n/H$, and the kernel of ρ_H is H.

Example 13.2. The coset space of $H = \mathfrak{D}_4$ has size $4!/8 = 3$, and a possible transversal is $\sigma_1 = e$, $\sigma_2 = (12)$, $\sigma_3 = (14)$. We have $\rho_H(\mathfrak{S}_4) \cong \mathfrak{S}_3$: indeed, the permutation $\sigma = (12) \in \mathfrak{S}_4$ is such that $(H\sigma_1\sigma, H\sigma_2\sigma, H\sigma_3\sigma) = (H\sigma_2, H\sigma_1, H\sigma_3)$, therefore $\rho_H(\sigma) = (12)$, and the permutation $\tilde{\sigma} = (14) \in \mathfrak{S}_4$ is such that $(H\sigma_1\tilde{\sigma}, H\sigma_2\tilde{\sigma}, H\sigma_3\tilde{\sigma}) = (H\sigma_3, H\sigma_2, H\sigma_1)$, therefore $\rho_H(\tilde{\sigma}) = (13)$. Thus $\rho_H(\mathfrak{S}_4)$ must be a subgroup of $\mathfrak{S}_3$ containing (12) and (13), that is, $\mathfrak{S}_3$ itself.

With the same technique, noting that $\sigma_i^{-1}\sigma_j$ maps the coset $H\sigma_i$ to $H\sigma_j$, we can prove that the action ρ_H of $\mathfrak{S}_n$ over the coset space is transitive.

Group action and composition of quasi-means

We may define a right action of $\mathfrak{S}_n$ on the set of quasi-means of n matrices as

$$(Q\sigma)(A_1, \ldots, A_n) := Q(A_{\sigma(1)}, \ldots, A_{\sigma(n)}).$$

The choice of putting σ to the right, albeit slightly unusual, was chosen to simplify some of the notations used in Section 13.5.

When Q is a quasi-mean of r matrices and $R_1, R_2, \ldots R_r$ are quasi-means of n matrices, let us define $Q \circ (R_1, R_2, \ldots R_r)$ as the map

$$(Q \circ (R_1, R_2, \ldots R_r))(\underline{A}) := Q(R_1(\underline{A}), R_2(\underline{A}), \ldots, R_r(\underline{A})). \tag{13.10}$$

Theorem 13.3 ([Pol09])**.** *Let $Q(A_1, \ldots, A_r)$ and $R_j(A_1, \ldots A_n)$ (for $j = 1, \ldots, r$) be quasi-means. Then,*

1. *For all $\sigma \in \mathfrak{S}_r$, $Q\sigma$ is a quasi-mean.*
2. *$(A_1, \ldots, A_r, A_{r+1}) \mapsto Q(A_1, \ldots, A_r)$ is a quasi-mean.*

3. $Q \circ (R_1, R_2, \dots R_r)$ *is a quasi-mean.*

Proof. All properties follow directly from the monotonicity (P4) and from the corresponding properties for the means Q and R_j. □

We may then define the *isotropy group*, or *stabilizer group* of a quasi-mean Q

$$\mathrm{Stab}(Q) := \{\sigma \in \mathfrak{S}^n : Q = Q\sigma\}. \tag{13.11}$$

13.4 Means obtained as map compositions

Let us define the concept of *reductive symmetries* of a quasi-mean as follows.

- in the special case in which $G_2(A, B) = A \# B$, the symmetry property that $A \# B = B \# A$ is a reductive symmetry.
- let $Q \circ (R_1, \dots, R_r)$ be a quasi-mean obtained by composition. The symmetry with respect to the permutation σ (i.e., the fact that $Q = Q\sigma$) is a *reductive symmetry* for $Q \circ (R_1, \dots, R_r)$ if this property can be formally proved relying only on the reductive symmetries of Q and $R_1, \dots, R_r$.

For instance, if we take $Q(A, B, C) := A \#(B \# C)$, then we can deduce that $Q(A, B, C) = Q(A, C, B)$ for all A, B, C, but not that $Q(A, B, C) = Q(B, C, A)$ for all A, B, C. This does not imply that such a symmetry property does not hold: if we were considering the operator $+$ instead of $\#$, then it would hold that $A + B + C = B + C + A$, but there are no means of proving it relying only on the commutativity of addition — in fact, associativity is crucial.

As we stated in the introduction, Ando, Li and Mathias [ALM04] showed explicit counterexamples proving that all the symmetry properties of G^{4rec} and G^{rec} are reductive symmetries. We conjecture the following.

Conjecture 1. All the symmetries of a quasi-mean obtained by recursive composition from G_2 are reductive symmetries.

In other words, we postulate that no "unexpected symmetries" appear while examining quasi-means compositions. This is a rather strong statement; however, the numerical experiments and the theoretical analysis performed up to now never showed any invariance property that could not be inferred by those of the underlying means.

We prove several result limiting the reductive symmetries that a mean can have; to this aim, we introduce the *reductive isotropy group*

$$\mathrm{RStab}(Q) = \{\sigma \in \mathrm{Stab}(Q) : Q = Q\sigma \text{ is a reductive symmetry}\}. \tag{13.12}$$

We prove in the following that there is no quasi-mean Q such that $\mathrm{RStab}(Q) = \mathfrak{S}_n$. This shows that the existing "tools" in the mathematician's "toolbox" do not allow one to construct a matrix geometric mean (with full proof) based only on map compositions; thus we need either to devise a completely new construction or to find a novel way to prove additional invariance properties involving map compositions.

The following results show that when looking for a reductive matrix geometric mean, i.e., a quasi-mean Q with $\mathrm{RStab}\, Q = \mathfrak{S}_n$, we may restrict our search to quasi-means of a special form.

Theorem 13.4 ([Pol09]). *Let Q be a quasi-mean of $r+s$ matrices, and $R_1, R_2, \dots, R_r$, $S_1, S_2, \dots, S_s$ be quasi-means of n matrices such that $R_i \neq S_j\sigma$ for all i, j and every $\sigma \in \mathfrak{S}_n$. Then,*

$$\operatorname{RStab}(Q \circ (R_1, R_2, \dots, R_r, S_1, S_2, \dots, S_s)) \subseteq \operatorname{RStab}(Q \circ (R_1, \dots, R_r, R_1, R_1, \dots, R_1)). \quad (13.13)$$

Proof. Let $\sigma \in \operatorname{RStab}(Q \circ (R_1, R_2, \dots, R_r, S_1, S_2, \dots S_s))$; since the only invariance properties that we may assume on Q are those predicted by its invariance group, it must be the case that

$$(R_1\sigma, R_2\sigma, \dots, R_r\sigma, S_1\sigma, S_2\sigma, \dots S_s\sigma)$$

is a permutation of $(R_1, R_2, \dots, R_r, S_1, S_2, \dots S_s)$ belonging to $\operatorname{RStab}(Q)$. Since $R_i \neq S_j\sigma$, this permutation must map the sets $\{R_1, R_2, \dots, R_r\}$ and $\{S_1, S_2, \dots S_s\}$ to themselves. Therefore, the same permutation maps

$$(R_1, R_2, \dots, R_r, R_1, R_1, \dots R_1)$$

to

$$(R_1\sigma, R_2\sigma, \dots, R_r\sigma, R_1\sigma, R_1\sigma, \dots R_1\sigma).$$

This implies that

$$Q(R_1, R_2, \dots, R_r, R_1, R_1, \dots R_1) = Q(R_1\sigma, R_2\sigma, \dots, R_r\sigma, R_1\sigma, R_1\sigma, \dots R_1\sigma)$$

as requested. □

Theorem 13.5 ([Pol09]). *Let $M_1 := Q \circ (R_1, R_2, \dots, R_r)$ be a quasi-mean. Then there is a quasi-mean M_2 in the form*

$$\tilde{Q} \circ (\tilde{R}\sigma_1, \tilde{R}\sigma_2, \dots, \tilde{R}\sigma_{\tilde{r}}), \quad (13.14)$$

where $(\sigma_1, \sigma_2, \dots, \sigma_{\tilde{r}})$ is a right coset transversal for $\operatorname{RStab}(\tilde{R})$ in $\mathfrak{S}_n$, such that $\operatorname{RStab}(M_1) \subseteq \operatorname{RStab}(M_2)$.

Proof. Set $\tilde{R} = R_1$. For each $i = 2, 3, \dots, r$ if $R_i \neq \tilde{R}\sigma$, we may replace it with $\tilde{R}$, and by Theorem 13.4 the restricted isotropy group increases or stays the same. Thus by repeated application of this theorem, we may reduce to the case in which each R_i is in the form $\tilde{R}\tau_i$ for some permutation τ_i.

Since $\{\sigma_i\}$ is a right transversal, we may write $\tau_i = h_i\sigma_{k(i)}$ for some $h_i \in H$ and $k(i) \in \{1, 2, \dots, \tilde{r}\}$. We have $\tilde{R}h = \tilde{R}$ since $h \in \operatorname{Stab}\tilde{R}$, thus $R_i = \tilde{R}\sigma_{k(i)}$. The resulting quasi-mean is $Q \circ (\tilde{R}\sigma_{k(1)}, \dots, \tilde{R}\sigma_{k(r)})$. Notice that we may have $k(i) = k(j)$, or some cosets may be missing. Let now $\tilde{Q}$ be defined as $\tilde{Q}(A_1, A_2, \dots, A_{\tilde{r}}) := Q(A_{k(1)}, \dots, A_{k(r)})$; then we have

$$\tilde{Q}(\tilde{R}\sigma_1, \dots, \tilde{R}\sigma_{\tilde{r}}) = Q(\tilde{R}\sigma_{k(1)}, \dots, \tilde{R}\sigma_{k(r)}) \quad (13.15)$$

and thus the isotropy groups of the left-hand side and right-hand side coincide. □

For the sake of brevity, we define

$$Q \circ R := Q \circ (R\sigma_1, \dots, R\sigma_r),$$

assuming a standard choice of the transversal for $H = \operatorname{Stab} R$. Notice that $Q \circ R$ depends on the ordering of the cosets $H\sigma_1, \dots, H\sigma_r$, but not on the choice of the coset representative σ_i, since $Qh\sigma_i = Q\sigma_i$ for each $h \in H$.

Example 13.6. The quasi-mean $(A,B,C)\mapsto (A\#B)\#(B\#C)$ is $Q\circ Q$, where $Q(X,Y,Z)=X\#Y$, $H=\{e,(12)\}$, and the transversal is $\{e,(13),(23)\}$.

Example 13.7. The quasi-mean $(A,B,C)\mapsto (A\#B)\#C$ is not in the form (13.14), but in view of Theorem 13.4, its restricted isotropy group is a subgroup of that of $(A,B,C)\mapsto (A\#B)\#(A\#B)$.

The following theorem shows which permutations we can actually prove to belong to the reductive isotropy group of a mean in the form (13.14).

Theorem 13.8 ([Pol09])**.** *Let $H\le\mathfrak{S}_n$, R be a quasi-mean of n matrices such that $\operatorname{RStab}R=H$ and Q be a quasi-mean of $r=n!/|H|$ matrices. Let $G\in\mathfrak{S}_n$ be the largest permutation subgroup such that $\rho_H(G)\le\operatorname{RStab}(Q)$. Then, $G=\operatorname{RStab}(Q\circ R)$.*

Proof. Let $\sigma\in G$ and $\tau=\rho_H(\sigma)$; we have

$$\begin{aligned}(Q\circ R)\sigma(\underline{A}) &= Q\big(R\sigma_1\sigma(\underline{A}),R\sigma_2\sigma(\underline{A}),\dots,R\sigma_r\sigma(\underline{A})\big)\\ &= Q\big(R\sigma_{\tau(1)}(\underline{A}),R\sigma_{\tau(2)}(\underline{A}),\dots,R\sigma_{\tau(r)}(\underline{A})\big)\\ &= Q\big(R\sigma_1(\underline{A}),R\sigma_2(\underline{A}),\dots,R\sigma_r(\underline{A})\big),\end{aligned}$$

where the last equality holds because $\tau\in\operatorname{Stab}(Q)$.

Notice that the above construction is the only way to obtain invariance with respect to a given permutation σ: indeed, to prove invariance relying only on the invariance properties of Q, $(R\sigma_1\sigma,\dots,R\sigma_r\sigma)=(R\sigma_{\tau(1)},\dots,R\sigma_{\tau(r)})$ must be a permutation of $(R\sigma_1,\dots,R\sigma_r)$ belonging to $\operatorname{RStab}Q$, and thus $\rho_H(\sigma)=\tau\in\operatorname{Stab}Q$. Thus the reductive invariance group of the composite mean is precisely the largest subgroup G such that $\rho_H(G)\le\operatorname{Stab}Q$. □

Example 13.9. Let $n=4$, Q be any (reductive) geometric mean of three matrices (i.e., $\operatorname{RStab}Q=\mathfrak{S}_3$), and $R(A,B,C,D):=(A\#C)\#(B\#D)$. We have $H=\operatorname{RStab}R=\mathfrak{D}_4$, the dihedral group over four elements, with cardinality 8. There are $r=4!/|H|=3$ cosets. Since $\rho_H(\mathfrak{S}_4)$ is a subset of $\operatorname{Stab}Q=\mathfrak{S}_3$, the isotropy group of $Q\circ R$ contains $G=\mathfrak{S}_4$ by Theorem 13.8. Therefore $Q\circ R$ is a geometric mean of four matrices.

Indeed, the only assertion we have to check explicitly is that $\operatorname{RStab}R=\mathfrak{D}_4$. The isotropy group of R contains (24) and (1234), since by using repeatedly the fact that $\#$ is symmetric in its arguments we can prove that $R(A,B,C,D)=R(A,D,C,B)$ and $R(A,B,C,D)=R(D,A,B,C)$. Thus it must contain the subgroup generated by these two elements, that is, $\mathfrak{D}_4\le\operatorname{RStab}R$. The only subgroups of $\mathfrak{S}_4$ containing $\mathfrak{D}_4$ as a subgroup are the two trivial ones $\mathfrak{S}_4$ and $\mathfrak{D}_4$. We cannot have $\operatorname{RStab}R=\mathfrak{S}_4$, since R has the same definition as G^{4rec} of equation (13.8), apart from a reordering, and it was proved [ALM04] that this is not a geometric mean.

It is important to notice that by choosing $G_3=G_3^{ALM}$ or $G_3=G_3^{BMP}$ in the previous example we may obtain a geometric mean of four matrices using a single limit process, the one needed for G_3. This is more efficient than G_4^{ALM} and G_4^{BMP}, which compute a mean of four matrices via several means of three matrices, each of which requires a limit process in its computation. We return to this topic in Section 13.6.

Is it possible to obtain a reductive geometric mean of n matrices, for $n>4$, starting from simpler means and using the construction of Theorem 13.8? The following result shows that the answer is no.

Theorem 13.10 ([Pol09])**.** *Suppose $G:=\operatorname{RStab}(Q\circ R)\ge\mathfrak{A}_n$ and $n>4$. Then $\mathfrak{A}_n\le\operatorname{RStab}(Q)$ or $\mathfrak{A}_n\le\operatorname{RStab}(R)$.*

Proof. Let us consider $K = \ker \rho_H$. It is a normal subgroup of $\mathfrak{S}_n$, but for $n > 4$ the only normal subgroups of $\mathfrak{S}_n$ are the trivial group $\{e\}$, $\mathfrak{A}_n$ and $\mathfrak{S}_n$ [DM96]. Let us consider the three cases separately.

1. $K = \{e\}$. In this case, $\rho_H(G) \cong G$, and thus $G \leq \operatorname{RStab} Q$.
2. $K = \mathfrak{S}_n$. In this case, $\rho_H(\mathfrak{S}_n)$ is the trivial group. But the action of $\mathfrak{S}_n$ over the coset space is transitive, since $\sigma_i^{-1}\sigma_j$ sends the coset $H\sigma_i$ to the coset $H\sigma_j$. So the only possibility is that there is a single coset in the coset space, i.e., $H = \mathfrak{S}_n$.
3. $K = \mathfrak{A}_n$. As in the above case, since the action is transitive, it must be the case that there are at most two cosets in the coset space, and thus $H = \mathfrak{S}_n$ or $H = \mathfrak{A}_n$. □

Thus it is impossible to apply Theorem 13.8 to obtain a quasi-mean with reductive isotropy group containing $\mathfrak{A}_n$, unless one of the two starting quasi-means has a reductive isotropy group already containing $\mathfrak{A}_n$.

13.5 Means obtained as limits

We describe now a unifying algebraic setting in terms of isotropy groups, generalizing the procedures leading to the means defined by limit processes G^{ALM}, G^{BMP} and G^{P}.

Let $S : (\mathbb{P}^m)^n \to (\mathbb{P}^m)^n$ be a map; we say that S *preserves* a subgroup $H < \mathfrak{S}_n$ if there is a map $\tau : H \to H$ such that $Sh(\underline{A}) = \tau(h)S(\underline{A})$ for all $\underline{A} \in \mathbb{P}^m$.

Theorem 13.11 ([Pol09])**.** *Let $S : (\mathbb{P}^m)^n \to (\mathbb{P}^m)^n$ be a map and $H < \mathfrak{S}_n$ be a permutation group such that*

1. *$(\underline{A}) \to \big(S(\underline{A})\big)_i$ is a quasi-mean for all $i = 1, \ldots, n$,*
2. *S preserves H,*
3. *for all $\underline{A} \in (\mathbb{P}^m)^n$, $\lim_{k\to\infty} S^k(\underline{A})$ is a scalar n-tuple[1],*

and let us denote by $S^\infty(\underline{A})$ the common value of all entries of the scalar n-tuple $\lim_{k\to\infty} S^k(\underline{A})$. Then, $S^\infty(\underline{A})$ is a quasi-mean with isotropy group containing H.

Proof. From Theorem 13.3, it follows that $\underline{A} \mapsto \big(S^k(\underline{A})\big)_i$ is a quasi-mean for each k. Since all the properties defining a quasi-mean pass to the limit, S^∞ is a quasi-mean itself.

Let us take $h \in H$ and $\underline{A} \in \mathbb{P}^n$. It is easy to prove by induction on k that $S^k h(\underline{A}) = \tau^k(h)\big(S^k(\underline{A})\big)$. Now, choose a matrix norm inducing the Euclidean topology on $\mathbb{P}^m$; let $\varepsilon > 0$ be fixed, and let us take K such that for all $k > K$ and for all $i = 1, \ldots, n$ the following inequalities hold:

- $\big\|\big(S^k(\underline{A})\big)_i - S^\infty(\underline{A})\big\| < \varepsilon$,
- $\big\|\big(S^k h(\underline{A})\big)_i - S^\infty h(\underline{A})\big\| < \varepsilon$.

We know that $\big(S^k h(\underline{A})\big)_i = \big(\tau^k(h) S^k(\underline{A})\big)_i = \big(S^k(\underline{A})\big)_{\tau^k(h)(i)}$, therefore

$$\|S^\infty(\underline{A}) - S^\infty h(\underline{A})\| \leq \Big\|\big(S^k(\underline{A})\big)_{\tau^k(h)(i)} - S^\infty(\underline{A})\Big\| + \big\|\big(S^k h(\underline{A})\big)_i - S^\infty h(\underline{A})\big\| < 2\varepsilon.$$

[1]Here S^k denotes function iteration: $S^1 = S$ and $S^{k+1}(\underline{A}) = S(S^k(\underline{A}))$ for all k.

Since ε is arbitrary, the two limits must coincide. This holds for each $h \in H$, therefore $H \leq \operatorname{Stab} S^\infty$. □

Example 13.12. The map S defining G_4^{ALM} is

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \mapsto \begin{bmatrix} G_3^{ALM}(B,C,D) \\ G_3^{ALM}(A,C,D) \\ G_3^{ALM}(A,B,D) \\ G_3^{ALM}(A,B,C) \end{bmatrix}.$$

One can see that $S\sigma = \sigma^{-1}S$ for each $\sigma \in \mathfrak{S}_4$, and thus with the choice $\tau(\sigma) := \sigma^{-1}$ we get that S preserves $\mathfrak{S}_4$. Thus, by Theorem 13.11, $S^\infty = G_4^{ALM}$ is a geometric mean of four matrices. The same reasoning applies to G^{BMP}.

Example 13.13. The map S defining G_4^P is

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \mapsto \begin{bmatrix} A \# B \\ B \# C \\ C \# D \\ D \# A \end{bmatrix}.$$

S preserves the dihedral group $\mathfrak{D}_4$. Therefore, provided the iteration process converges to a scalar n-tuple, S^∞ is a quasi-mean with isotropy group containing $\mathfrak{D}_4$.

As in the previous section, we are interested in seeing whether this approach, which is the one that has been used to prove invariance properties of the known limit means [ALM04, BMP10a], can yield better results for a different map S.

Theorem 13.14 ([Pol09])**.** *Let $S : (\mathbb{P}^m)^n \to (\mathbb{P}^m)^n$ preserve a group H. Then, the invariance group of each of its components S_i, $i = 1, \ldots, n$, is a subgroup of H of index at most n.*

Proof. Let i be fixed, and set $I_k := \{h \in H : \tau(h)(i) = k\}$. The sets I_k are mutually disjoint and their union is H, so the largest one has cardinality at least $|H|/n$, let us call it $I_{\overline{k}}$.

From the hypothesis that S preserves H, we get $S_i h(\underline{A}) = S_{\overline{k}}(\underline{A})$ for each $\underline{A}$ and each $h \in I_{\overline{k}}$. Let g be an element of $I_{\overline{k}}$; then $S_i h(g^{-1}\underline{A}) = S_{\overline{k}}(g^{-1}\underline{A}) = S_i(\underline{A})$. Thus the isotropy group of S_i contains all the elements of the form hg^{-1}, $h \in I_{\overline{k}}$, and those are at least $|H|/n$. □

The following result holds.

Theorem 13.15 ([DM96, page 147])**.** *For $n > 4$, the only subgroups of $\mathfrak{S}_n$ with index at most n are:*

- *the alternating group $\mathfrak{A}_n$,*
- *the n groups $T_k = \{\sigma \in \mathfrak{S}_n : \sigma(k) = k\}$, $k = 1, \ldots, n$, all of which are isomorphic to $\mathfrak{S}_{n-1}$,*
- *for $n = 6$ only, another conjugacy class of 6 subgroups of index 6 isomorphic to $\mathfrak{S}_5$.*

Analogously, the only subgroups of $\mathfrak{A}_n$ with index at most n are:

- *the n groups $U_k = \{\sigma \in \mathfrak{A}_n : \sigma(k) = k\}$, $k = 1, \ldots, n$, all of which are isomorphic to $\mathfrak{A}_{n-1}$,*

- *for $n = 6$ only, another conjugacy class of* 6 *subgroups of index* 6 *isomorphic to* $\mathfrak{A}_5$.

This shows that whenever we try to construct a geometric mean of n matrices by taking a limit processes, such as in the Ando–Li–Mathias approach, the isotropy groups of the starting means must contain $\mathfrak{A}_{n-1}$. On the other hand, by Theorem 13.8, we cannot generate means whose isotropy group contains $\mathfrak{A}_{n-1}$ by composition of simpler means; therefore, there is no simpler approach than that of building a mean of n matrices as a limit process of means of $n-1$ matrices (or at least quasi-means with $\operatorname{Stab} Q = \mathfrak{A}_{n-1}$, which makes little difference). This shows that the recursive approach of G^{ALM} and G^{BMP} cannot be simplified while still maintaining P3 (permutation invariance).

13.6 Computational issues and numerical experiments

The results we have exposed up to now are negative results, and they hold for $n > 4$. On the other hand, it turns out that for $n = 4$, since $\mathfrak{A}_n$ is not a simple group, there is the possibility of obtaining a mean that is computationally simpler than the ones in use. Such a mean is the one we described in Example 13.9. Let us take any mean of three elements (we use G_3^{BMP} here since it is the one with the best computational results); the new mean is therefore defined as

$$G_4^{NEW}(A,B,C,D) := G_3^{BMP}\left((A\#B)\#(C\#D), (A\#C)\#(B\#D), (A\#D)\#(B\#C)\right). \quad (13.16)$$

Notice that only one limit process is needed to compute the mean; conversely, when computing G_4^{ALM} or G_4^{BMP} we are performing an iteration whose elements are computed by doing four additional limit processes; thus we may expect a large saving in the overall computational cost.

We may extend the definition recursively to $n > 4$ elements using the construction described in (13.6), but with G^{NEW} instead of G^{BMP}. The total computational cost, computed in the same fashion as for the ALM and BMP means, is $O(n!p_3p_5p_6\dots p_n m^3)$. Thus the undesirable dependence from $n!$ does not disappear; the new mean should only yield a saving measured by a multiplicative constant in the complexity bound.

We have implemented the original BMP algorithm and the new one described in the above section with Matlab and run some tests on the same set of examples used by Moakher [Moa06] and Bini *et al.* [BMP10a]. It is an example deriving from physical experiments on elasticity. It consists of five sets of matrices to average, with n varying from 4 to 6, and 6×6 matrices split into smaller diagonal blocks.

For each of the five data sets, we have computed both the BMP and the new matrix mean. The CPU times are reported in Table 13.1. As a stopping criterion for the iterations, we used

$$\max_{i,j,k} \left| (A_i^{(h)})_{jk} - (A_i^{(h+1)})_{jk} \right| < 10^{-13}.$$

As we expected, our mean provides a substantial reduction of the CPU time which is roughly by an order of magnitude.

Following Bini *et al.* [BMP10a], we then focused on the second data set (ammonium dihydrogen phosphate) for a deeper analysis; we report in Table 13.6 the number of iterations and matrix roots needed in both computations.

Data set (number of matrices)	BMP mean	New mean
$NaClO_3$ (5)	$1.3 \times 10^{+00}$	3.1×10^{-01}
Ammonium dihydrogen phosphate (4)	3.5×10^{-01}	3.9×10^{-02}
Potassium dihydrogen phosphate (4)	3.5×10^{-01}	3.9×10^{-02}
Quartz (6)	$2.9 \times 10^{+01}$	$6.7 \times 10^{+00}$
Rochelle salt (4)	6.0×10^{-01}	5.5×10^{-02}

Table 13.1: CPU times for the elasticity data sets

	BMP mean	New mean
Outer iterations ($n = 4$)	3	none
Inner iterations ($n = 3$)	2.0 (avg.)	2
Matrix square roots (`sqrtm`)	72	15
Matrix p-th roots (`rootm`)	84	6

Table 13.2: Number of inner and outer iterations needed, and number of matrix roots needed (ammonium dihydrogen phosphate)

	BMP mean	New mean
Outer iterations ($n = 4$)	4	none
Inner iterations ($n = 3$)	2.5 (avg.)	3
Matrix square roots (`sqrtm`)	120	18
Matrix p-th roots (`rootm`)	136	9

Table 13.3: Number of inner and outer iterations needed, and number of matrix roots needed (on matrices (13.17))

The examples in these data sets are mainly composed of matrices very close to each other; we consider here instead an example of mean of four matrices whose mutual distances are larger:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \quad C = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \quad D = \begin{bmatrix} 20 & 0 & -10 \\ 0 & 20 & 0 \\ -10 & 0 & 20 \end{bmatrix}. \tag{13.17}$$

The results regarding these matrices are reported in Table 13.3.

Accuracy

It is not clear how to check the accuracy of a limit process yielding a matrix geometric mean, since the exact value of the mean is not known *a priori*, except for the cases in which all the A_i commute. In those cases, P1 yields a compact expression for the result. So we cannot test accuracy in the general case; instead, we have focused on two special examples.

As a first accuracy experiment, we computed $G(M^{-2}, M, M^2, M^3) - M$, where M is taken as the first matrix of the second data set on elasticity; the result of this computation should be zero according to P1. As a second experiment, we tested the validity of P9

Operation	Result
$\left\|G^{BMP}(M^{-2}, M, M^2, M^3) - M\right\|_2$	4.0×10^{-14}
$\left\|G^{NEW}(M^{-2}, M, M^2, M^3) - M\right\|_2$	2.5×10^{-14}
$\left\|\det(G^{BMP}(A, B, C, D)) - (\det(A)\det(B)\det(C)\det(D))^{1/4}\right\|$	5.5×10^{-13}
$\left\|\det(G^{BMP}(A, B, C, D)) - (\det(A)\det(B)\det(C)\det(D))^{1/4}\right\|$	2.1×10^{-13}

Table 13.4: Accuracy tests

(determinant identity) on the means of the four matrices in (13.17). The results of both computations are reported in Table 13.4; the results are well within the errors permitted by the stopping criterion, and show that both algorithms can reach a satisfying precision.

13.7 Conclusions and research lines

The results of this chapter show that, by combining existing matrix means, it is possible to create a new mean which is faster to compute than the existing ones. Moreover, we show that using only function compositions and limit processes with the existing proof strategies, it is not possible to achieve any further significant improvement with respect to the existing algorithms. In particular, the dependency from $n!$ cannot be removed. New attempts should focus on other aspects, such as:

- finding new "unexpected" algebraic relationships involving the existing matrix means, which would allow to break out of the framework of Theorem 13.8–Theorem 13.11; or, on the other hand, proving that there are no such unexpected algebraic relations (Conjecture 1).
- introducing new kinds of matrix geometric means or quasi-means, different from the ones built using function composition and limits.
- proving that the Riemannian centroid (13.1) is a matrix mean in the sense of Ando–Li–Mathias (currently P4 is an open problem), and providing faster and reliable algorithms to compute it.

It is an interesting question in itself whether it is possible to construct a quasi-mean whose isotropy group is exactly $\mathfrak{A}_n$.

Chapter 14

Conclusions

We have presented a unitary introduction to a number of matrix equations appearing in applications and in mathematical literature, and some old and new algorithms for their solution. The relationships between the different quadratic vector and matrix equations are partially exploited to give better algorithms and unified proofs. However, some details still cannot be embedded elegantly in the theory, and many algorithms for one equation of the class have no counterpart for the others.

Among the iterative algorithms for matrix equations, an important role is played by the Structured Doubling Algorithm. We explored the connection between SDA and cyclic reduction and introduced a couple of implementation tricks that could help in implementing effectively this algorithm and adapting it to other equations and problems.

Moreover, two auxiliary problems related to quadratic matrix equations are considered: the structure preservation in displacement rank-structured Riccati equations, and the problem of extending the matrix geometric mean to several matrices.

We linked the solution of the NARE appearing in neutron transport to the theory of Cauchy-like matrices; recognizing and exploiting their computational properties has a great impact in the numerical solution of this equation, and is a necessary step in all algorithms that aim to reach a reasonable computational efficiency.

The problem of finding a sensible generalization of the matrix geometric mean is mathematically fascinating; we introduced some construction that lead to more computationally effective matrix means satisfying all the required properties, and presented a negative result that shows that the existing techniques cannot be pushed further.

Many research problems are still open in the area of quadratic matrix equations; we hope that the results described in this thesis helped casting some light on the mathematical picture behind the solution algorithms, and we look forward to exploring the research lines that were briefly summarized at the end of each chapter.

Acknowledgments

Some of the original research results described in this thesis have been produced with the help of several coauthors, to whom goes the author's gratitude.

* * *

With this thesis, the author is attaining his terminal degree, at the end of a long course of studies. He is grateful to a number of mentors and teachers, starting from his parents, that have led him along these years through his formation, to scientific studies, to mathematics, and to numerical linear algebra.

The light I shine today
is what you gave to me

* * *

In addition, he acknowledges the company of many friends, colleagues, companions, mates, people sharing an interest with him, and passing acquaintances that could have become closer. Feel free to insert your name here.

The love I have renounced
can never be replaced

Bibliography

[ALM04] T. Ando, Chi-Kwong Li, and Roy Mathias. Geometric means. *Linear Algebra Appl.*, 385:305–334, 2004. doi:10.1016/j.laa.2003.11.019. Cited on pages 141, 142, 143, 145, 152, 155, 156, 157, 158, 160, 162, and 164.

[AN68] Brian D. O. Anderson and R. W. Newcomb. Impedance synthesis via state-space techniques. *Proc. IEE*, 115(7):928–936, 1968. Cited on page 117.

[AN04] K. B. Athreya and P. E. Ney. *Branching processes*. Dover Publications Inc., Mineola, NY, 2004. Reprint of the 1972 original [Springer, New York; MR0373040]. Cited on page 27.

[And66] Brian D. O. Anderson. Algebraic description of bounded real matrixes. *Electronics Letters*, 2(12):464–465, 1966. doi:10.1049/el:19660392. Cited on page 117.

[And78] Brian D. O. Anderson. Second-order convergent algorithms for the steady-state Riccati equation. *Internat. J. Control*, 28(2):295–306, 1978. Cited on pages 2, 57, 58, and 63.

[AR04] Soohan Ahn and V. Ramaswami. Transient analysis of fluid flow models via stochastic coupling to a queue. *Stoch. Models*, 20(1):71–101, 2004. doi:10.1081/STM-120028392. Cited on page 45.

[AR09] Antonio Aricò and Giuseppe Rodriguez. A fast solver for linear systems with displacement structure. submitted for publication, 2009. Available from: http://tex.unica.it/~arico/publications/drsolve.pdf. Cited on pages 87, 89, and 104.

[Asm95] Søren Asmussen. Stationary distributions for fluid flow models with or without Brownian noise. *Comm. Statist. Stochastic Models*, 11(1):21–49, 1995. Cited on page 45.

[AV73] Brian D. O. Anderson and Sumeth Vongpanitlerd. *Network analysis and synthesis: a modern systems theory approach*. Prentice-Hall Englewood Cliffs, N.J.,, 1973. Cited on page 117.

[Bar07] N. E. Barabanov. Kalman-Yakubovich lemma in general finite dimensional case. *Internat. J. Robust Nonlinear Control*, 17(5-6):369–386, 2007. doi:10.1002/rnc.1162. Cited on pages 117 and 118.

[BB98] Peter Benner and Ralph Byers. An exact line search method for solving generalized continuous-time algebraic Riccati equations. *IEEE Trans. Automat. Control*, 43(1):101–107, 1998. doi:10.1109/9.654908. Cited on page 53.

[BB06] Peter Benner and Ralph Byers. An arithmetic for matrix pencils: theory and new algorithms. *Numer. Math.*, 103(4):539–573, 2006. doi:10.1007/s00211-006-0001-x. Cited on pages 54, 56, 57, and 62.

[BD93] Zhaojun Bai and James W. Demmel. On swapping diagonal blocks in real Schur form. *Linear Algebra Appl.*, 186:73–95, 1993. doi:10.1016/0024-3795(93)90286-W. Cited on page 52.

[BDG97] Zhaojun Bai, James Demmel, and Ming Gu. An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems. *Numer. Math.*, 76(3):279–308, 1997. doi:10.1007/s002110050264. Cited on pages 58 and 62.

[Ben97] Peter Benner. *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*. Doctoral dissertation, Fakultät fr Mathematik, TU Chemnitz-Zwickau, Chemnitz, 1997. Published by Logos-Verlag, Berlin. Cited on pages 57, 58, and 117.

[BGL08] Zhong-Zhi Bai, Yong-Hua Gao, and Lin-Zhang Lu. Fast iterative schemes for non-symmetric algebraic Riccati equations arising from transport theory. *SIAM J. Sci. Comput.*, 30(2):804–818, 2008. doi:10.1137/060675344. Cited on page 20.

[BGM02] D. A. Bini, L. Gemignani, and B. Meini. Computations with infinite Toeplitz matrices and polynomials. *Linear Algebra Appl.*, 343/344:21–61, 2002. Special issue on structured and infinite systems of linear equations. doi:10.1016/S0024-3795(01)00341-X. Cited on page 42.

[BGN70] B. L. Buzbee, G. H. Golub, and C. W. Nielson. On direct methods for solving Poisson's equations. *SIAM J. Numer. Anal.*, 7:627–656, 1970. Cited on pages 2 and 41.

[BH06a] Rajendra Bhatia and John Holbrook. Noncommutative geometric means. *Math. Intelligencer*, 28(1):32–39, 2006. doi:10.1007/BF02987000. Cited on pages 141 and 144.

[BH06b] Rajendra Bhatia and John Holbrook. Riemannian geometry and matrix geometric means. *Linear Algebra Appl.*, 413(2-3):594–618, 2006. doi:10.1016/j.laa.2005.08.025. Cited on pages 141, 155, and 156.

[Bha07] Rajendra Bhatia. *Positive definite matrices*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2007. Cited on pages 4, 141, 155, and 156.

[BILM06] Dario A. Bini, Bruno Iannazzo, Guy Latouche, and Beatrice Meini. On the solution of algebraic Riccati equations arising in fluid queues. *Linear Algebra Appl.*, 413(2-3):474–494, 2006. doi:10.1016/j.laa.2005.04.019. Cited on pages 45, 63, 68, and 111.

[BIMP10] Dario A. Bini, Bruno Iannazzo, Beatrice Meini, and Federico Poloni. Nonsymmetric algebraic riccati equations associated with an m-matrix: recent advances and algorithms. In Vadim Olshevsky and Eugene Tyrtyshnikov, editors, *Numerical Methods for Structured Markov Chains*, pages 176–209. World Scientific Publishing Company, 2010. Available from: http://drops.dagstuhl.de/opus/volltexte/2008/1395. Cited on pages 6, 42, 47, 50, 54, 62, and 64.

[BIP08] Dario A. Bini, Bruno Iannazzo, and Federico Poloni. A fast Newton's method for a nonsymmetric algebraic Riccati equation. *SIAM J. Matrix Anal. Appl.*, 30(1):276–290, 2008. doi:10.1137/070681478. Cited on pages 22, 45, 85, 92, 106, 110, and 114.

[BKM05] Peter Benner, Daniel Kressner, and Volker Mehrmann. Skew-Hamiltonian and Hamiltonian eigenvalue problems: theory, algorithms and applications. In *Proceedings of the Conference on Applied Mathematics and Scientific Computing*, pages 3–39, Dordrecht, 2005. Springer. doi:10.1007/1-4020-3197-1_1. Cited on page 122.

[BKO02] Tibor Boros, Thomas Kailath, and Vadim Olshevsky. Pivoting and backward stability of fast algorithms for solving Cauchy linear equations. *Linear Algebra Appl.*, 343/344:63–99, 2002. Special issue on structured and infinite systems of linear equations. doi:10.1016/S0024-3795(01)00519-5. Cited on page 98.

[BKT08] Nigel G. Bean, Nectarios Kontoleon, and Peter G. Taylor. Markovian trees: properties and algorithms. *Ann. Oper. Res.*, 160:31–50, 2008. doi:10.1007/s10479-007-0295-9. Cited on pages 1, 16, 20, and 35.

[BLM95a] Peter Benner, Alan Laub, and Volker Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: the continuous-time case. Technical Report SPC 95-22, Forschergruppe 'Scientific Parallel Computing', Fakultät für Mathematik, TU Chemnitz-Zwickau, 1995. Available from: http://www.tu-chemnitz.de/sfb393/Files/PS/spc95_22.ps.gz. Cited on pages 123, 133, and 134.

[BLM95b] Peter Benner, Alan Laub, and Volker Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations II: the discrete-time case. Technical Report SPC 95-23, Forschergruppe 'Scientific Parallel Computing', Fakultät für Mathematik, TU Chemnitz-Zwickau, 1995. Available from: http://www.tu-chemnitz.de/sfb393/Files/PS/spc95_23.ps.gz. Cited on page 133.

[BLM03] Dario A. Bini, Guy Latouche, and Beatrice Meini. Solving nonlinear matrix equations arising in tree-like stochastic processes. *Linear Algebra Appl.*, 366:39–64, 2003. Special issue on structured matrices: analysis, algorithms and applications (Cortona, 2000). doi:10.1016/S0024-3795(02)00593-1. Cited on page 25.

[BLM05] D. A. Bini, G. Latouche, and B. Meini. *Numerical methods for structured Markov chains*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2005. Oxford Science Publications. doi:10.1093/acprof:oso/9780198527688.001.0001. Cited on pages 2, 16, 20, 39, 40, 41, 42, 63, and 69.

[BM96] Dario Bini and Beatrice Meini. On the solution of a nonlinear matrix equation arising in queueing problems. *SIAM J. Matrix Anal. Appl.*, 17(4):906–926, 1996. doi:10.1137/S0895479895284804. Cited on pages 41 and 42.

[BM09] Dario A. Bini and Beatrice Meini. The cyclic reduction algorithm: from Poisson equation to stochastic processes and beyond. In memoriam of Gene H. Golub. *Numer. Algorithms*, 51(1):23–60, 2009. doi:10.1007/s11075-008-9253-0. Cited on pages 42 and 63.

[BMP09] Dario A. Bini, Beatrice Meini, and Federico Poloni. Fast solution of a certain Riccati equation through Cauchy-like matrices. *Electron. Trans. Numer. Anal.*, 33:84–104, 2008/09. Cited on pages 3, 92, 95, and 114.

[BMP10a] Dario A. Bini, Beatrice Meini, and Federico Poloni. An effective matrix geometric mean satisfying the Ando-Li-Mathias properties. *Math. Comp.*, 79(269):437–452, 2010. doi:10.1090/S0025-5718-09-02261-3. Cited on pages 145, 147, 155, 156, 157, 164, and 165.

[BMP10b] Dario A. Bini, Beatrice Meini, and Federico Poloni. Transforming algebraic Riccati equations into unilateral quadratic matrix equations. *Numer. Math.*, 2010. To appear in print. doi:10.1007/s00211-010-0319-2. Cited on pages 45, 63, 65, 66, 70, 71, 73, and 75.

[BMR08] Dario A. Bini, Beatrice Meini, and Vaidyanathan Ramaswami. A probabilistic interpretation of cyclic reduction and its relationships with logarithmic reduction. *Calcolo*, 45(3):207–216, 2008. doi:10.1007/s10092-008-0151-6. Cited on page 63.

[BOT05] Nigel G. Bean, Małgorzata M. O'Reilly, and Peter G. Taylor. Algorithms for return probabilities for stochastic fluid flows. *Stoch. Models*, 21(1):149–184, 2005. doi:10.1081/STM-200046511. Cited on pages 2, 27, 45, 47, 63, 64, and 78.

[BP94a] Abraham Berman and Robert J. Plemmons. *Nonnegative matrices in the mathematical sciences*, volume 9 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994. Revised reprint of the 1979 original. Cited on page 6.

[BP94b] Dario Bini and Victor Y. Pan. *Polynomial and matrix computations. Vol. 1*. Progress in Theoretical Computer Science. Birkhäuser Boston Inc., Boston, MA, 1994. Fundamental algorithms. Cited on page 93.

[BS72] R. H. Bartels and G. W. Stewart. Solution of the matrix equation $AX + XB = C$. *Commun. ACM*, 15(9):820–826, 1972. doi:10.1145/361573.361582. Cited on page 53.

[CA77] David J. Clements and Brian D. O. Anderson. Transformational solution of singular linear-quadratic control problems. *IEEE Trans. Automatic Control*, AC-22(1):57–60, 1977. Cited on page 117.

[CA78] David J. Clements and Brian D. O. Anderson. *Singular optimal control: the linear-quadratic problem*. Springer-Verlag, Berlin, 1978. Lecture Notes in Control and Information Sciences, Vol. 5. Cited on page 117.

[CALM97] D. J. Clements, Brian D. O. Anderson, A. J. Laub, and J. B. Matson. Spectral factorization with imaginary-axis zeros. *Linear Algebra Appl.*, 250:225–252, 1997. doi:10.1016/0024-3795(95)00525-0. Cited on pages 117 and 118.

[CAM77] David J. Clements, Brian D. O. Anderson, and Peter J. Moylan. Matrix inequality solution to linear-quadratic singular control problems. *IEEE Trans. Automatic Control*, AC-22(1):55–57, 1977. Cited on page 117.

[CCG+09] Chun-Yueh Chiang, Eric King-Wah Chu, Chun-Hua Guo, Tsung-Ming Huang, Wen-Wei Lin, and Shu-Fang Xu. Convergence analysis of the doubling algorithm for several nonlinear matrix equations in the critical case. *SIAM J. Matrix Anal. Appl.*, 31(2):227–247, 2009. doi:10.1137/080717304. Cited on pages 42, 58, 59, 63, and 130.

[CFL05] E. K.-W. Chu, H.-Y. Fan, and W.-W. Lin. A structure-preserving doubling algorithm for continuous-time algebraic Riccati equations. *Linear Algebra Appl.*, 396:55–80, 2005. doi:10.1016/j.laa.2004.10.010. Cited on pages 3, 58, 59, 122, and 123.

[CG89] D. J. Clements and K. Glover. Spectral factorization via Hermitian pencils. *Linear Algebra Appl.*, 122/123/124:797–846, 1989. doi:10.1016/0024-3795(89)90677-0. Cited on pages 117 and 127.

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Advances in Cryptology EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, chapter 27, pages 392–407. Springer Berlin Heidelberg, Berlin, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_27. Cited on page 26.

[CW95] Xin Chen and John T. Wen. Positive realness preserving model reduction with $\mathcal{H}_\infty$ norm error bounds. *IEEE Trans. Circuits Systems I Fund. Theory Appl.*, 42(1):23–29, 1995. doi:10.1109/81.350793. Cited on page 117.

[Cyb80] George Cybenko. The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations. *SIAM J. Sci. Statist. Comput.*, 1(3):303–319, 1980. doi:10.1137/0901021. Cited on page 83.

[DDH07] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numer. Math.*, 108(1):59–91, 2007. doi:10.1007/s00211-007-0114-x. Cited on pages 62 and 135.

[DM96] John D. Dixon and Brian Mortimer. *Permutation groups*, volume 163 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1996. Cited on pages 163 and 164.

[dSSL02] Ana da Silva Soares and Guy Latouche. Further results on the similarity between fluid queues and QBDs. In *Matrix-analytic methods (Adelaide, 2002)*, pages 89–106. World Sci. Publ., River Edge, NJ, 2002. Cited on page 45.

[Fas00] Heike Fassbender. *Symplectic methods for the symplectic eigenproblem*. Kluwer Academic/Plenum Publishers, New York, 2000. Cited on page 125.

[FD87] Bruce A. Francis and John C. Doyle. Linear control theory with an H_∞ optimality criterion. *SIAM J. Control Optim.*, 25(4):815–844, 1987. doi:10.1137/0325046. Cited on page 127.

[FG06] Sandra Fital and Chun-Hua Guo. Convergence of the solution of a nonsymmetric matrix Riccati differential equation to its stable equilibrium solution. *J. Math. Anal. Appl.*, 318(2):648–657, 2006. doi:10.1016/j.jmaa.2005.06.040. Cited on pages 45, 46, and 50.

[FLPR99] Matteo Frigo, Charles E. Leiserson, Harald Prokop, and Sridhar Ramachandran. Cache-oblivious algorithms. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 285, Washington, DC, USA, 1999. IEEE Computer Society. doi:10.1109/SFFCS.1999.814600. Cited on page 92.

[GA04] Serkan Gugercin and Athanasios C. Antoulas. A survey of model reduction by balanced truncation and some new results. *Internat. J. Control*, 77(8):748–766, 2004. doi:10.1080/00207170410001713448. Cited on page 117.

[Gan98] F. R. Gantmacher. *The theory of matrices. Vol. 1*. AMS Chelsea Publishing, Providence, RI, 1998. Translated from the Russian by K. A. Hirsch, Reprint of the 1959 translation. Cited on page 8.

[Ger88] A. Gerasoulis. A fast algorithm for the multiplication of generalized Hilbert matrices with vectors. *Math. Comp.*, 50(181):179–188, 1988. doi:10.2307/2007921. Cited on pages 84, 85, 92, and 114.

[GH06] Chun-Hua Guo and Nicholas J. Higham. A Schur-Newton method for the matrix pth root and its inverse. *SIAM J. Matrix Anal. Appl.*, 28(3):788–804 (electronic), 2006. doi:10.1137/050643374. Cited on pages 45, 49, 51, 52, 53, and 111.

[GH07] Chun-Hua Guo and Nicholas J. Higham. Iterative solution of a nonsymmetric algebraic Riccati equation. *SIAM J. Matrix Anal. Appl.*, 29(2):396–412, 2007. doi:10.1137/050647669. Cited on page 51.

[GIM07] Chun-Hua Guo, Bruno Iannazzo, and Beatrice Meini. On the doubling algorithm for a (shifted) nonsymmetric algebraic Riccati equation. *SIAM J. Matrix Anal. Appl.*, 29(4):1083–1100, 2007. doi:10.1137/060660837. Cited on pages 45, 46, 48, 49, 51, 54, 62, 64, 74, 75, 76, 106, and 111.

[GKO95] I. Gohberg, T. Kailath, and V. Olshevsky. Fast Gaussian elimination with partial pivoting for matrices with displacement structure. *Math. Comp.*, 64(212):1557–1576, 1995. doi:10.2307/2153371. Cited on pages 83, 85, 86, 98, and 99.

[GL00] Chun-Hua Guo and Alan J. Laub. On the iterative solution of a class of nonsymmetric algebraic Riccati equations. *SIAM J. Matrix Anal. Appl.*, 22(2):376–391 (electronic), 2000. doi:10.1137/S089547989834980X. Cited on pages 16, 17, 20, 21, 45, 53, 106, and 112.

[GLR82] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix polynomials*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1982. Computer Science and Applied Mathematics. Cited on page 7.

[GLR06] Israel Gohberg, Peter Lancaster, and Leiba Rodman. *Invariant subspaces of matrices with applications*, volume 51 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006. Reprint of the 1986 original. Cited on pages 124 and 133.

[GLX06] Xiao-Xia Guo, Wen-Wei Lin, and Shu-Fang Xu. A structure-preserving doubling algorithm for nonsymmetric algebraic Riccati equation. *Numer. Math.*, 103(3):393–412, 2006. doi:10.1007/s00211-005-0673-7. Cited on pages 2, 61, 63, 68, 75, and 76.

[GO83] A. Griewank and M. R. Osborne. Analysis of Newton's method at irregular singularities. *SIAM J. Numer. Anal.*, 20(4):747–773, 1983. doi:10.1137/0720050. Cited on page 7.

[Gu98] Ming Gu. Stable and efficient algorithms for structured systems of linear equations. *SIAM J. Matrix Anal. Appl.*, 19(2):279–306 (electronic), 1998. doi:10.1137/S0895479895291273. Cited on page 100.

[Guo01] Chun-Hua Guo. Nonsymmetric algebraic Riccati equations and Wiener-Hopf factorization for M-matrices. *SIAM J. Matrix Anal. Appl.*, 23(1):225–242 (electronic), 2001. doi:10.1137/S0895479800375680. Cited on pages 18, 21, 45, 46, 47, 48, 49, 53, 64, 75, and 105.

[Guo02] Chun-Hua Guo. A note on the minimal nonnegative solution of a nonsymmetric algebraic Riccati equation. *Linear Algebra Appl.*, 357:299–302, 2002. doi:10.1016/S0024-3795(02)00431-7. Cited on pages 45, 46, and 49.

[Guo06] Chun-Hua Guo. Efficient methods for solving a nonsymmetric algebraic Riccati equation arising in stochastic fluid models. *J. Comput. Appl. Math.*, 192(2):353–373, 2006. doi:10.1016/j.cam.2005.05.012. Cited on pages 45, 47, 48, 52, 69, and 106.

[Guo07] Chun-Hua Guo. A new class of nonsymmetric algebraic Riccati equations. *Linear Algebra Appl.*, 426(2-3):636–649, 2007. doi:10.1016/j.laa.2007.05.044. Cited on pages 64 and 74.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996. Cited on pages 6, 48, 52, 112, 123, and 131.

[HC92] Per Christian Hansen and Tony F. Chan. Fortran subroutines for general Toeplitz systems. *ACM Trans. Math. Softw.*, 18(3):256–273, 1992. doi:10.1145/131766.131768. Cited on pages 83 and 100.

[HCL05] T.-M. Hwang, E. K.-W. Chu, and W.-W. Lin. A generalized structure-preserving doubling algorithm for generalized discrete-time algebraic Riccati equations. *Internat. J. Control*, 78(14):1063–1075, 2005. doi:10.1080/00207170500155827. Cited on pages 45 and 122.

[Hea52] R. F. S. Hearmon. The elastic constants of piesoelectric crystals. *Br. J. Appl. Phys.*, 3(4):120–123, 1952. doi:10.1088/0508-3443/3/4/303. Cited on page 152.

[Higa] Nicholas J. Higham. The Matrix Computation Toolbox. Available from: `http://www.ma.man.ac.uk/~higham/mctoolbox`. Cited on page 152.

[Higb] Nicholas J. Higham. The Matrix Function Toolbox. Available from: `http://www.ma.man.ac.uk/~higham/mftoolbox`. Cited on page 53.

[Hig08] Nicholas J. Higham. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. Theory and computation. Cited on pages 10, 11, 53, 54, 62, and 124.

[HJ94] Roger A. Horn and Charles R. Johnson. *Topics in matrix analysis*. Cambridge University Press, Cambridge, 1994. Corrected reprint of the 1991 original. Cited on page 67.

[HL09] Tsung-Ming Huang and Wen-Wei Lin. Structured doubling algorithms for weakly stabilizing Hermitian solutions of algebraic Riccati equations. *Linear Algebra Appl.*, 430(5-6):1452–1478, 2009. `doi:10.1016/j.laa.2007.08.043`. Cited on pages 58, 59, 125, and 127.

[HLR08] Sophie Hautphenne, Guy Latouche, and Marie-Ange Remiche. Newton's iteration for the extinction probability of a Markovian binary tree. *Linear Algebra Appl.*, 428(11-12):2791–2804, 2008. `doi:10.1016/j.laa.2007.12.024`. Cited on pages 1, 16, 20, 21, 28, and 33.

[HLR09] Sophie Hautphenne, Guy Latouche, and Marie-Ange Remiche. Algorithmic approach to the extinction probability of branching processes. *Methodology and Computing in Applied Probability*, 2009. To appear in print. `doi:10.1007/s11009-009-9141-7`. Cited on page 33.

[HMR02] C. He, B. Meini, and N. H. Rhee. A shifted cyclic reduction algorithm for quasi-birth-death problems. *SIAM J. Matrix Anal. Appl.*, 23(3):673–691 (electronic), 2001/02. `doi:10.1137/S0895479800371955`. Cited on pages 62, 106, and 111.

[Hoc65] R. W. Hockney. A fast direct solution of Poisson's equation using Fourier analysis. *J. Assoc. Comput. Mach.*, 12:95–113, 1965. Cited on page 41.

[Hog07] Leslie Hogben, editor. *Handbook of linear algebra*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2007. Associate editors: Richard Brualdi, Anne Greenbaum and Roy Mathias. Cited on pages 47, 49, and 59.

[HP03] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. Cited on page 87.

[HR84] Georg Heinig and Karla Rost. *Algebraic methods for Toeplitz-like matrices and operators*, volume 13 of *Operator Theory: Advances and Applications*. Birkhäuser Verlag, Basel, 1984. Cited on page 92.

[HSC06] Roger A. Horn and Stefano Serra-Capizzano. A general setting for the parametric Google matrix. *Internet Math.*, 3(4):385–411, 2006. Cited on page 48.

[HVH10] Sophie Hautphenne and Benny Van Houdt. On the link between Markovian trees and tree-structured Markov chains. *European J. Oper. Res.*, 201(3):791–798, 2010. `doi:10.1016/j.ejor.2009.03.052`. Cited on page 21.

[IB03] Bruno Iannazzo and Dario Bini. A cyclic reduction method for solving algebraic Riccati equations. Technical report, Dipartimento di Matematica, Università di Pisa, 2003. Cited on page 63.

[IM09] Bruno Iannazzo and Beatrice Meini. Computing the matrix geometric mean: a unifying framework. Technical report, Dipartimento di Matematica, Università di Pisa, 2009. Cited on page 141.

[IP] Bruno Iannazzo and Federico Poloni. The worst-case scenario in nonlinear matrix equations. To appear in the proceedings of NSMC 2010 — numerical solution of Markov chains, Williamsburg, VA. Cited on page 62.

[JC93] Jonq 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(1):65–80, 1993. doi:10.1080/00411459308203530. Cited on page 20.

[JL99] Jonq Juang and Wen-Wei Lin. Nonsymmetric algebraic Riccati equations and Hamiltonian-like matrices. *SIAM J. Matrix Anal. Appl.*, 20(1):228–243 (electronic), 1999. doi:10.1137/S0895479897318253. Cited on pages 3, 45, and 105.

[JS71] D. H. Jacobson and Jason L. Speyer. Necessary and sufficient conditions for optimality for singular control problems: A limit approach. *J. Math. Anal. Appl.*, 34:239–266, 1971. Cited on page 117.

[Jua01] Jonq Juang. Global existence and stability of solutions of matrix Riccati equations. *J. Math. Anal. Appl.*, 258(1):1–12, 2001. doi:10.1006/jmaa.2000.7058. Cited on pages 18 and 45.

[Kan52] L. V. Kantorovich. *Functional analysis and applied mathematics.* NBS Rep. 1509. U. S. Department of Commerce National Bureau of Standards, Los Angeles, Calif., 1952. Translated by C. D. Benster. Cited on page 53.

[KC94] T. Kailath and J. Chun. Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices. *SIAM J. Matrix Anal. Appl.*, 15(1):114–128, 1994. doi:10.1137/S0895479889169042. Cited on pages 84 and 87.

[KKT06] Michael Karow, Daniel Kressner, and Françoise Tisseur. Structured eigenvalue condition numbers. *SIAM J. Matrix Anal. Appl.*, 28(4):1052–1068 (electronic), 2006. doi:10.1137/050628519. Cited on page 122.

[Kle68] D. Kleinman. On an iterative technique for riccati equation computations. *IEEE Trans. Automat. Control*, 13(1):114–115, 1968. Cited on page 53.

[KO97] Thomas Kailath and Vadim Olshevsky. Diagonal pivoting for partially reconstructible Cauchy-like matrices, with applications to Toeplitz-like linear equations and to boundary rational matrix interpolation problems. In *Proceedings of the Fifth Conference of the International Linear Algebra Society (Atlanta, GA, 1995)*, volume 254, pages 251–302, 1997. doi:10.1016/S0024-3795(96)00288-1. Cited on pages 85, 86, 92, and 93.

[KS95] Thomas Kailath and Ali H. Sayed. Displacement structure: theory and applications. *SIAM Rev.*, 37(3):297–386, 1995. doi:10.1137/1037082. Cited on page 85.

[KVZ+72] M. A. Krasnosel′skiĭ, G. M. Vaĭnikko, P. P. Zabreĭko, Ya. B. Rutitskii, and V. Ya. Stetsenko. *Approximate solution of operator equations.* Wolters-Noordhoff Publishing, Groningen, 1972. Translated from the Russian by D. Louvish. Cited on page 17.

[Lau79] Alan J. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Trans. Automat. Control*, 24(6):913–921, 1979. doi:10.1109/TAC.1979.1102178. Cited on page 52.

[Lim08] Yongdo Lim. On Ando-Li-Mathias geometric mean equations. *Linear Algebra Appl.*, 428(8-9):1767–1777, 2008. Cited on pages 141 and 155.

[LR80] P. Lancaster and L. Rodman. Existence and uniqueness theorems for the algebraic Riccati equation. *Internat. J. Control*, 32(2):285–309, 1980. doi:10.1080/00207178008922858. Cited on page 63.

[LR93] Guy Latouche and V. Ramaswami. A logarithmic reduction algorithm for quasi-birth-death processes. *J. Appl. Probab.*, 30(3):650–674, 1993. Cited on pages 2, 40, and 69.

[LR95] Peter Lancaster and Leiba Rodman. *Algebraic Riccati equations*. Oxford Science Publications. The Clarendon Press Oxford University Press, New York, 1995. Cited on pages 2, 12, 26, 46, 64, 68, and 117.

[Lu05a] Lin-Zhang Lu. Newton iterations for a non-symmetric algebraic Riccati equation. *Numer. Linear Algebra Appl.*, 12(2-3):191–200, 2005. doi:10.1002/nla.411. Cited on pages 16, 21, 45, 106, 109, 112, and 113.

[Lu05b] Lin-Zhang Lu. Solution form and simple iteration of a nonsymmetric algebraic Riccati equation arising in transport theory. *SIAM J. Matrix Anal. Appl.*, 26(3):679–685 (electronic), 2005. doi:10.1137/S0895479801397275. Cited on pages 3, 16, 18, 20, 45, 105, and 106.

[Lur51] A. I. Lur′e. *Nekotorye nelineĭnye zadači teorii avtomatičeskogo regulirovaniya (Certain Nonlinear Problems in the Theory of Automatic Control)*. Gosudarstv. Izdat. Tehn.-Teor. Lit., Moscow-Leningrad, 1951. Translated into English, H.M. Stationery, 1957. Cited on page 117.

[LX06] Wen-Wei Lin and Shu-Fang Xu. Convergence analysis of structure-preserving doubling algorithms for Riccati-type matrix equations. *SIAM J. Matrix Anal. Appl.*, 28(1):26–39 (electronic), 2006. doi:10.1137/040617650. Cited on pages 45, 58, 59, and 74.

[LY08] David G. Luenberger and Yinyu Ye. *Linear and nonlinear programming*. International Series in Operations Research & Management Science, 116. Springer, New York, third edition, 2008. Cited on page 123.

[Mal89] A. N. Malyshev. Calculation of invariant subspaces of a regular linear matrix pencil. *Sibirsk. Mat. Zh.*, 30(4):76–86, 217, 1989. doi:10.1007/BF00971756. Cited on pages 57 and 62.

[Meh91] Volker 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. doi:10.1007/BFb0039443. Cited on pages 3, 64, and 124.

[Moa05] Maher Moakher. A differential geometric approach to the geometric mean of symmetric positive-definite matrices. *SIAM J. Matrix Anal. Appl.*, 26(3):735–747 (electronic), 2005. doi:10.1137/S0895479803436937. Cited on page 155.

[Moa06] Maher Moakher. On the averaging of symmetric positive-definite tensors. *J. Elasticity*, 82(3):273–296, 2006. doi:10.1007/s10659-005-9035-z. Cited on pages 4, 141, 152, 155, 156, and 165.

[MP] Volker Mehrmann and Federico Poloni. A generalized structured doubling algorithm for control problems in invariant subspace form. In preparation. Cited on pages 55, 56, 130, 131, 132, 133, and 135.

[MP09] Andrej Muhič and Bor Plestenjak. On the singular two-parameter eigenvalue problem. *Electron. J. Linear Algebra*, 18:420–437, 2009. Cited on page 25.

[MP10] Beatrice Meini and Federico Poloni. A Perron iteration for the solution of a quadratic vector equation arising in Markovian binary trees, 2010. Submitted for publication. arXiv:arXiv:1006.0577. Cited on pages 30, 31, 32, and 38.

[MRT05] P. G. Martinsson, V. Rokhlin, and M. Tygert. A fast algorithm for the inversion of general Toeplitz matrices. *Comput. Math. Appl.*, 50(5-6):741–752, 2005. doi:10.1016/j.camwa.2005.03.011. Cited on page 114.

[MS88] Carl D. Meyer and G. W. Stewart. Derivatives and perturbations of eigenvectors. *SIAM J. Numer. Anal.*, 25(3):679–691, 1988. doi:10.1137/0725041. Cited on page 30.

[OJ88] Philippe C. Opdenacker and Edmond A. Jonckheere. A contraction mapping preserving balanced reduction scheme and its infinity norm error bounds. *IEEE Trans. Circuits and Systems*, 35(2):184–189, 1988. doi:10.1109/31.1720. Cited on page 117.

[Ols03] Vadim Olshevsky. Pivoting for structured matrices and rational tangential interpolation. In *Fast algorithms for structured matrices: theory and applications (South Hadley, MA, 2001)*, volume 323 of *Contemp. Math.*, pages 1–73. Amer. Math. Soc., Providence, RI, 2003. Cited on page 100.

[OR00] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*, volume 30 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Reprint of the 1970 original. Cited on pages 7, 15, and 44.

[Pál09] Miklós Pálfia. The Riemann barycenter computation and means of several matrices. *Int. J. Comput. Math. Sci.*, 3(3):128–133, 2009. Cited on pages 155, 156, and 157.

[PDS02] Joel Phillips, Luca Daniel, and L. Miguel Silveira. Guaranteed passive balancing transformations for model order reduction. In *DAC '02: Proceedings of the 39th annual Design Automation Conference*, pages 52–57, New York, NY, USA, 2002. ACM. doi:10.1145/513918.513933. Cited on page 117.

[Pol09] Federico Poloni. Constructing matrix geometric means, 2009. To appear in Electron. J. Linear Algebra. arXiv:arXiv:0906.3132. Cited on pages 158, 159, 161, 162, 163, and 164.

[Pol10a] Federico Poloni. A note on the $O(n)$-storage implementation of the gko algorithm and its adaptation to trummer-like matrices. *Numer. Algorithms*, 2010. To appear in print. arXiv:arXiv:0903.4569, doi:10.1007/s11075-010-9361-5. Cited on pages 90, 95, 97, 98, and 103.

[Pol10b] Federico Poloni. Quadratic vector equations, 2010. arXiv:arXiv:1004.1500. Cited on pages 16, 17, 18, 19, 20, 23, 25, and 43.

[PR] Federico Poloni and Timo Reis. A structured doubling algorithm for the numerical solution of Lur'e equations. In preparation. Cited on pages 60 and 125.

[Ram99] V. Ramaswami. Matrix analytic methods for stochastic fluid flows. In D. Smith and P. Hey, editors, *Teletraffic Engineering in a Competitive World*, volume 3 of *Proceedings of the 16th International Teletraffic Congress, Elsevier Science B.V., Edimburgh, UK*, pages 1019–1030, 1999. Cited on pages 3, 45, 63, 64, 65, 68, and 69.

[Rei09] Timo Reis. Lur'e equations and even matrix pencils. Technical Report 09-672, DFG Research Center MATHEON, Berlin, 2009. Submitted for publication. Available from: http://www.matheon.de/preprints/6757_lure_prepr.pdf. Cited on pages 118, 119, and 121.

[Rob80] J. D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Internat. J. Control*, 32(4):677–687, 1980. doi:10.1080/00207178008922881. Cited on page 57.

[Rod06] G. Rodriguez. Fast solution of Toeplitz- and Cauchy-like least-squares problems. *SIAM J. Matrix Anal. Appl.*, 28(3):724–748 (electronic), 2006. doi:10.1137/050629148. Cited on pages 87 and 104.

[Rog94] L. C. G. Rogers. Fluid models in queueing theory and Wiener-Hopf factorization of Markov chains. *Ann. Appl. Probab.*, 4(2):390–413, 1994. Available from: http://links.jstor.org/sici?sici=1050-5164(199405)4:2<390:FMIQTA>2.0.CO;2-0&origin=MSN. Cited on page 45.

[RR02] André C. M. Ran and Martine C. B. Reurings. The symmetric linear matrix equation. *Electron. J. Linear Algebra*, 9:93–107 (electronic), 2002. Cited on page 26.

[RS08] Timo Reis and Tatjana Stykel. Balanced truncation for electrical circuits. Preprint 2008/32, Institut für Mathematik, TU Berlin, 2008. Submitted for publication. Available from: http://www.math.tu-berlin.de/~stykel/Publications/pr_08_32.pdf. Cited on page 117.

[RS10] Timo Reis and Tatjana Stykel. Positive real and bounded real balancing for model reduction of descriptor systems. *Internat. J. Control*, 83(1):74–88, 2010. doi:10.1080/00207170903100214. Cited on page 117.

[SB95] D. R. Sweet and R. P. Brent. Error analysis of a fast partial pivoting method for structured matrices. In *Proceedings SPIE, Advanced Signal Processing Algorithms SPIE*, volume 2563, pages 266–280, 1995. doi:10.1117/12.211404. Cited on pages 98, 99, 100, and 112.

[SJ71] Jason L. Speyer and David H. Jacobson. Necessary and sufficient conditions for optimality for singular control problems; a transformation approach. *J. Math. Anal. Appl.*, 33:163–187, 1971. Cited on page 117.

[Son98] Eduardo D. Sontag. *Mathematical control theory*, volume 6 of *Texts in Applied Mathematics*. Springer-Verlag, New York, second edition, 1998. Deterministic finite-dimensional systems. Cited on pages 3 and 11.

[SQO04] Xiaobai Sun and Enrique S. Quintana-Ortí. Spectral division methods for block generalized Schur decompositions. *Math. Comp.*, 73(248):1827–1847 (electronic), 2004. doi:10.1090/S0025-5718-04-01667-9. Cited on page 55.

[Tho76] R. C. Thompson. The characteristic polynomial of a principal subpencil of a Hermitian matrix pencil. *Linear Algebra and Appl.*, 14(2):135–177, 1976. Cited on page 9.

[Tre87] Harry L. Trentelman. Families of linear-quadratic problems: continuity properties. *IEEE Trans. Automat. Control*, 32(4):323–329, 1987. doi:10.1109/TAC.1987.1104593. Cited on page 117.

[Tsa00] Michael J. Tsatsomeros. Principal pivot transforms: properties and applications. *Linear Algebra Appl.*, 307(1-3):151–165, 2000. doi:10.1016/S0024-3795(99)00281-5. Cited on page 61.

[VBHK01] Marc Van Barel, Georg Heinig, and Peter Kravanja. A stabilized superfast solver for nonsymmetric Toeplitz systems. *SIAM J. Matrix Anal. Appl.*, 23(2):494–510 (electronic), 2001. doi:10.1137/S0895479899362302. Cited on page 83.

[Wal96] James S. Walker. *Fast Fourier transforms*. Studies in Advanced Mathematics. CRC Press, Boca Raton, FL, second edition, 1996. With 1 IBM-PC floppy disk (3.5 inch; HD). Cited on page 92.

[Wil71] Jan C. Willems. Least squares stationary optimal control and the algebraic Riccati equation. *IEEE Trans. Automatic Control*, AC-16:621–634, 1971. Cited on page 117.

[Wil72] Jan C. Willems. Dissipative dynamical systems. II. Linear systems with quadratic supply rates. *Arch. Rational Mech. Anal.*, 45:352–393, 1972. Cited on page 117.

[Wil82] David 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.*, pages 91–94. Springer, Berlin, 1982. Cited on page 45.

[Wil88] J. H. Wilkinson. *The algebraic eigenvalue problem*. Monographs on Numerical Analysis. The Clarendon Press Oxford University Press, New York, 1988. Oxford Science Publications. Cited on pages 30 and 131.

[WWS94] H. Weiss, Q. Wang, and J. L. Speyer. System characterization of positive real conditions. *IEEE Trans. Automat. Control*, 39(3):540–544, 1994. doi:10.1109/9.280753. Cited on page 117.

[Yak85] V. A. Yakubovich. A singular problem of optimal control of a linear steady state system with a quadratic functional. *Sibirsk. Mat. Zh.*, 26(1):189–200, 1985. Cited on page 117.

[ZDG96] Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. Prentice Hall, 1996. Cited on page 117.

Index