Ústav Teorie Informace a Automatizace
Akademie Věd České Republiky

Kandidátská disertace

Reliable Algorithms for Polynomial Matrices

Aspirant Ing. Didier Henrion

Školitel Ing. Michael Šebek, DrSc.

Vědní obor 26-15-9 technická kybernetika

Rok vydání 1998

Dizertace k získání vědecké hodnosti kandidáta technických věd

Uchazeč: Ing. Didier Henrion, externí aspirant ÚTIA AV ČR, pracovník LAAS-CNRS, Toulouse, Francie

Oponentí: Prof. Ing. Jiří Gregor, DrSc., FEL ČVUT, Praha
Ing. Jan Ježek, CSc., ÚTIA AV ČR, Praha
Ing. Jaroslav Doležal, CSc., Honeywell, Praha

- Dizertační práce byla vypracována v ÚTIA AV ČR v Praze
- Stanovisko k dizertaci vypracoval ÚTIA AV ČR v Praze
- Autoreferát byl rozeslán dne 30.9.1998
- Obhajoba dizertace se konala dne 22.12.1998 v 10.00 hod. před komisi pro obhajoby kandidátských dizertačních prací v oboru 26-15-9 Technická kybernetika v Ústavu teorie informace a automatizace AV ČR, Praha 8, Pod vodárenskou věží 4, v místnosti č. 203

Abstract

A unified approach is pursued for designing efficient and numerically reliable algorithms for polynomial matrices. Based on two standard polynomial techniques – Sylvester matrices and interpolation – several algorithms are developed for performing fundamental operations such as polynomial matrix rank evaluation, polynomial matrix triangularization (as a special case of greatest common divisor extraction) and resolution of bilateral symmetric matrix polynomial equations arising in continuous- and discrete-time spectral factorization. Relevant features of this unified approach are efficiency and reliability, as illustrated by numerical experiments. The algorithms were implemented and tested as a part of the Polynomial Toolbox for MATLAB.

Anotace

Práce přináší jednotný přístup k návrhu účinných a numericky spolehlivých algoritmů pro polynomiální matice. Na základě standardních polynomiálních technik – Sylvestrovy matice a interpolace – je odvozeno několik nových algoritmů pro základní operace s polynomialními maticemi jako je výpočet hodnosti, trojúhelníkování (a tedy výpočet největších společných dělitelů) a řešení dvoustranných maticových rovnic, které jsou potřeba pro spojitou a diskrétní spektrální faktorizaci. Účinnost a spolehlivost nových postupů je ilustrována experimentálně. Algoritmy jsou implementovány a testovány v rámci souboru programů Polynomial Toolbox for MATLAB.

Résumé

Une approche unifiée est adoptée pour développer des algorithmes efficaces et numériquement fiables de manipulation de matrices polynomiales. En se basant sur deux techniques polynomiales classiques – les matrices de Sylvester et l'interpolation – plusieurs algorithmes sont conçus pour effectuer des opérations fondamentales comme l'évaluation du rang d'une matrice polynomiale, la triangularisation d'une matrice polynomiale (en tant que cas particulier d'extraction du plus grand commun diviseur) et la résolution d'équations matricielles polynomiales bilatérales symétriques utilisées dans le cadre de la factorisation spectrale en temps continu ou discret. Les caractéristiques notables de cette approche unifiée sont l'efficacité et la fiabilité, comme le montrent les tests numériques. Les algorithmes furent implémentés et testés dans le cadre de la Polynomial Toolbox for MATLAB.

Mojí milované Janě

Acknowledgments

First, I would like to thank my advisor Michael Šebek for his guidance and support and for allowing me to pursue my ideas. His consistent advice and encouragement were essential for completion of this thesis, and is highly appreciated.

I am greatly indebted to Vladimír Kučera for his confidence and support. Special thanks to Miroslava Součková and Jarmila Zoltánová for their valuable help and kind assistance during the time I have been at ÚTIA, Prague. I am also grateful to my friend Rudolf Rázek for his mastering of French at key moments of my life.

Thanks to Jiří Gregor, Jan Ježek and Jaroslav Doležal for carefully reading and reviewing the manuscript and serving on my thesis committee.

I would also like to thank Sophie Tarbouriech, Germain Garcia and Jacques Bernussou for providing me the opportunity of completing this thesis at LAAS-CNRS, Toulouse. I express my sincere thanks to my friends Jamal Daafouz, Denis Arzelier and Olivier Bachelier for many interesting discussions in a cheerful working environment.

But most of all, I am grateful to my parents and to my wife Jana for their love and support. Thank you for your understanding and for always being there for me.

Contents

Značení (Notations)

The following notations will be used throughout the manuscript.

$A(z)$ stands for a polynomial or polynomial matrix in the indeterminate z. The dependence on z is sometimes dropped for notational ease. By convention, we set $z = s$ for continuous-time systems and $z = d$ for discrete-time systems.

$\mathbb{R}$ stands for the field of real numbers, $\mathbb{R}[z]$ for the ring of polynomials of z with coefficients in $\mathbb{R}$ and $\mathbb{R}(z)$ for the field of rational fractions of z with coefficients in $\mathbb{R}$.

$\mathbb{C}$ stands for the field of complex numbers and $\mathbb{C}[z]$ for the ring of polynomials of z with coefficients in $\mathbb{C}$.

$\deg A(z)$ is the degree of a polynomial matrix $A(z)$, i.e. the highest degree occurring in $A(z)$.

The notations $\mathrm{row}_i A$, $\mathrm{col}_j A$ and $\mathrm{ent}_{i,j} A$ respectively stand for the i-th row, j-th column and i, j-th entry of a matrix A.

$\otimes$ stand for the Kronecker product and $\mathrm{vec} A$ for the column vector obtained by stacking the columns of matrix A.

$\mathbf{I}_n$ is the identity matrix of dimension n and $\mathbf{0}_{m\times n}$ is an $m \times n$ matrix with zero entries. The dimension may be omitted when guessed from the context.

Chapter 1

Přehled současného stavu problematiky (Present state of knowledge)

Since the seventies and the seminal work of Kučera [27], a sheer volume of literature has been devoted to the polynomial approach to system control, lending new insights into fundamental analysis and design issues. The polynomial (or algebraic) approach reformulates most of the ideas of the classical frequency-domain and state-space approaches into the algebra of polynomials. In broad terms, the line of attack of polynomial methods consists of establishing the equivalence between solving control problems and associated matrix polynomial equations. The technique is conceptually simple and reduces the synthesis procedure to solving linear and quadratic matrix polynomial equations. Several chapters have been devoted to this approach in standard textbooks on linear systems [27, 22, 10, 4, 11, 31]. It has been shown that almost all the control problems at hand are amenable to this kind of treatment, and significant results have been achieved that illustrate the power and elegance of the method. Recently, it has been recognized that several of the algebraic concepts – originally introduced in [27] for discrete linear time-invariant systems – can be generalized and extended to continuous time-varying systems [49], uncertain systems [26], non-linear or infinite-dimensional systems [34] and non-linear control design [1].

Polynomials and polynomial matrices, as fundamental components at the core of the polynomial approach, are unsurprisingly ubiquitous in linear systems and control theory. Polynomial matrix fractions describe multivariable linear systems [22]. Polynomial matrices are useful in the analysis of descriptor systems [29] and arise naturally in behavioral systems theory [52]. Frequency-domain solutions of control and filtering problems, such as stabilization by feedback and pole assignment [27], $\mathcal{H}_2$-optimal control [18], $\mathcal{H}_\infty$-optimal control [35, 36] or robust pole placement [26] involve manipulations of polynomial matrices when it comes to concrete solutions. Moreover, the study of polynomial matrices usually provides deep insights into the structure of systems. Concepts such as degree minimality, properness or divisibility are actually intimately related to inherent physical properties.

As a matter of fact, polynomial matrices play a key role in the realm of control theory.

During the last two decades, major breakthroughs in numerical linear algebra and the always increasing availability of more powerful computational tools have mostly been exploited by control engineers pursing the state-space approach to system control. Significant developments were achieved for solving the algebraic Riccati equation, a quadratic matrix equation frequently arising in optimal control problems [41], and more recently, for solving the more general linear matrix inequality optimization problems [9]. Researchers generally focused more on the formulation of control problems in terms of matrix polynomial equations rather than on numerical methods for solving these equations. However, it is known that deep similarities exist between the state-space and polynomial approaches [31]. As a consequence, numerical algorithms for polynomials, and, by association, polynomial matrices, do not have a good reputation for reliability and stability and state-space algorithms for control problems are much more in favor.

In the control community, polynomial equations are traditionally solved through the standard Euclidean division algorithm for polynomials [27], or also through the polynomial reduction algorithm, a sophisticated version of the Euclidean algorithm [19, 21]. These numerical methods are generally referred to as elementary polynomial operations. People working with computer implementations of these algorithms quickly realized that, in spite of low computational load and memory requirements, elementary polynomial operations may behave erratically. They usually return fairly accurate results, but sometimes can fail. The reasons behind this unreliable behavior are still subject to investigation, but very little has been done for overcoming these difficulties. It needs to be recognized that numerical analysis of polynomial matrices has received far less attention so far than numerical linear algebra, hence the disproportion between theoretical developments in polynomial control and their computation implementation. Up to the author's knowledge, there is no rigorous and unified theory on numerical properties of polynomial operations. As a consequence, the design of numerically reliable methods that may replace elementary polynomial operations has naturally turned into a challenging and promising area of research.

Chapter 2

Cíle disertace (Objectives)

The first objective of this thesis is to draw a state of the art of numerical techniques dealing with polynomial matrices. First, we underline the importance of polynomial matrices in control theory and propose a fairly complete survey of matrix polynomial equations arising in control problems. Note that we restrict ourselves to equations that are linear in the unknowns. Indeed, only a few works have been devoted so far to general quadratic matrix polynomial equations. Actually, quadratic equations are not well understood even from the theoretical point of view, thus precluding the development of sufficiently general resolution techniques. Then, we shortly describe already existing algorithms for solving linear matrix polynomial equations. As pointed out above, the most common approaches are based on elementary polynomial operations. We do not dwell upon existing algorithms so long, the interested reader being referred to the literature for a more rigorous treatment. Numerical properties are investigated for each of these approaches in very general terms: numerical reliability, time and memory requirements. Based on this, the lack of sufficiently general, reliable and efficient algorithms is underlined.

The second objective of this thesis is then to fulfill the need for such reliable algorithms. This has represented the most part of our work. Our algorithms are mostly designed for overcoming numerical instability. We describe in considerable detail how four key matrix polynomial problems have successfully been approached through our techniques: rank evaluation, triangularization and resolution of a symmetric bilateral equation in continuous- and discrete-time settings.

Chapter 3

Metody zpracování (Methods)

Our research has been proceeding along two thrusts: Sylvester matrices and interpolation, two popular and well-understood polynomial techniques. As may be expected, we shall draw on many classical results since these two techniques have already been exploited for a substantial amount of time by linear algebra theoreticians. However, the novelty in our approach is in that we use these tools in a systematic and unifying fashion that simultaneously ensures numerical reliability and computational efficiency. The key observation is that both Sylvester matrix and interpolation formulations of linear matrix polynomial equations lead to the resolution of a linear system of equations. As a result, algorithms based on these techniques only hinge upon well-worked and widely available numerical linear algebra tools, in turn usually based on stable orthogonal transformations.

3.1 Sylvester matrix approach

Suppose we are to solve the matrix polynomial equation

$$A(z)X(z) = B(z) \tag{3.1}$$

where $A(z)$ is a given $m \times n$ polynomial matrix, $B(z)$ is a given $m \times p$ polynomial matrix and $X(z)$ is an $n \times p$ polynomial matrix to be found. Write the polynomial matrices in terms of increasing powers of the indeterminate z, i.e.

$$\begin{array}{rcl} A(z) & = & A_0 + zA_1 + \ldots + z^{d_A} A_{d_A} \\ B(z) & = & B_0 + zB_1 + \ldots + z^{d_B} B_{d_B} \\ X(z) & = & X_0 + zX_1 + \ldots + z^{d_X} X_{d_X}, \end{array}$$

where $d_A = \deg A(z)$, $d_B = \deg B(z)$ and $d_X = \deg X(z)$. It is assumed that $d_A + d_X = d_B$. It the equality does not hold, then some leading matrix coefficients in $A(z)$ or $B(z)$ may be zero. By equating the powers of z in the above matrix polynomial equation, we can build an equivalent linear system of equations

$$\underbrace{\begin{bmatrix} A_0 & & & \mathbf{0} \\ A_1 & A_0 & & \\ \vdots & A_1 & \ddots & \\ A_{d_A} & \vdots & & A_0 \\ & A_{d_A} & & A_1 \\ & & \ddots & \vdots \\ \mathbf{0} & & & A_{d_A} \end{bmatrix}}_{\langle A \rangle_{d_X}} \underbrace{\begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{d_X} \end{bmatrix}}_{\overline{X}} = \underbrace{\begin{bmatrix} B_0 \\ B_1 \\ \vdots \\ B_{d_B} \end{bmatrix}}_{\overline{B}}. \tag{3.2}$$

Matrix $\langle A \rangle_{d_X}$ is referred to as the Sylvester matrix of $A(z)$ of order d_X. It has $(d_A + d_X + 1)m$ rows and $(d_X + 1)n$ columns. Since matrix polynomial equation (3.1) and linear system (3.2) are equivalent, solving (3.2) for a constant solution $\overline{X}$ amounts to solving (3.1) for a polynomial solution $X(z)$.

The problematic point in this approach is clearly the estimation of d_X, the degree of the solution $X(z)$. This a key parameter influencing the size of the Sylvester matrix, thus the number of equations and unknowns in the equivalent linear system of equations (3.2). We shall see that, in some cases, an upper bound on d_X can be found. In other cases, some assumptions on matrices $A(z)$ and $B(z)$ will guarantee the existence of a solution $X(z)$ of a given degree.

In [46], the authors proved that an important number of control problems can be addressed through the Sylvester matrix approach. Several sections in standard textbooks [22, 4, 11] also tend to illustrate that this conceptually simple technique often proves useful and may even lend new insights into fundamental control problems.

3.2 Interpolation approach

Now suppose that input polynomial matrices $A(z)$, $B(z)$ in equation (3.1) are only known through constant values $A(z_i)$, $B(z_i)$ taken at given distinct points z_i, $i = 0, 1, \ldots, d$ of the complex plane. Matrix polynomial equation (3.1) can be written

$$[z^{d_X} A(z) \ \cdots \ zA(z) \quad A(z)]\overline{X} = B(z).$$

Upon collecting the $d+1$ interpolated values, we obtain the linear system of equations

$$\underbrace{\begin{bmatrix} z_d^{d_X} A(z_d) & \cdots & z_d A(z_d) & A(z_d) \\ \vdots & & & \vdots \\ z_1^{d_X} A(z_1) & \cdots & z_1 A(z_1) & A(z_1) \\ z_0^{d_X} A(z_0) & \cdots & z_0 A(z_0) & A(z_0) \end{bmatrix}}_{\mathbf{L}_d} \overline{X} = \begin{bmatrix} B(z_d) \\ \vdots \\ B(z_1) \\ B(z_0) \end{bmatrix}. \tag{3.3}$$

Interpolated matrix $\mathbf{L}_d$ has $(d+1)m$ rows and $(d_X+1)n$ columns. We can easily check that $\mathbf{L}_d$ is actually the Sylvester matrix of $A(z)$ of order d_X multiplied to the left by a block Vandermonde matrix, viz.

$$\mathbf{L}_d = (M_d \otimes \mathbf{I}_m)\langle A\rangle_{d_X}$$

where

$$M_d = \begin{bmatrix} z_d^d & \cdots & z_d & 1 \\ \vdots & & & \vdots \\ z_1^d & \cdots & z_1 & 1 \\ z_0^d & \cdots & z_0 & 1 \end{bmatrix}.$$

Note that Vandermonde matrix M_d is invertible since points z_i are supposed to be distinct [15]. As a result, when $d = d_A + d_X$, matrix polynomial equation (3.1) and linear systems (3.2) and (3.3) are equivalent. Therefore, solving (3.3) for a constant solution $\overline{X}$ amounts to solving (3.1) for a polynomial solution $X(z)$.

The problematic point in this approach is the estimation of the number of interpolation points. This a key parameter influencing the size of the interpolated matrix, hence the number of equations and unknowns in the equivalent linear system of equations (3.3). We shall see that, in some cases, an upper bound on q can be found. In other cases, some assumptions on matrices $A(z)$ and $B(z)$ will guarantee the existence of a solution $X(z)$ for a given number of interpolation points.

In the survey paper [2], it has been shown that the interpolation approach supplies solutions to the most part of control problems involving polynomial matrices. Interpolation techniques also provide, in an elegant and flexible way, alternative interpretations of well-known linear system theory results. Applications include the resolution of key control problems such as pole assignment by state or output feedback [54], J-spectral factorization [37], or also $\mathcal{H}_\infty$ formulation of the optimal control problem [35]. The above are just a few of the many examples of the strong presence of interpolation results in the system and control literature.

3.3 Kronecker product

Finally, we also mention here a technical tool that will be used throughout the manuscript: the Kronecker product and its associated vector operator.

Let G, H and X denote matrices of suitable dimensions. The stacking operator vec and the Kronecker product $\otimes$ verify

$$\mathrm{vec}[GXH] = (H^T \otimes G)\mathrm{vec}X.$$

The constant square matrix P such that

$$\mathrm{vec}X^T = P\mathrm{vec}X$$

is called permutation matrix and can be used to commute the Kronecker product since

$$P^T(G \otimes H)P = H \otimes G.$$

Note that P is involutory ($P^2 = \mathbf{I}_{n^2}$), symmetric and therefore orthogonal ($P^T P = \mathbf{I}_{n^2}$). For details the reader is referred to [40, Chapter 12].

Chapter 4

Výsledky disertace s uvedením nových poznatků (New results)

As pointed out above, the first objective of our work is to provide a survey of numerical techniques dealing with polynomial matrices. For this purpose, in Section 4.1 we underline the importance of polynomial matrices in control by exposing a rather comprehensive list of problems involving matrix polynomial equations. In Section 4.2, we shortly describe existing algorithms for solving these equations. We particularly insist on numerical aspects and show that there is still a need for sufficiently general, reliable and efficient algorithms.

Then, we comply with our second objective which is the development of reliable numerical methods for solving matrix polynomial equations. We keep with our main impetus, which is the development of sufficiently general methods that fulfill efficiency and reliability requirements. All the methods presented in this chapter are believed to be new and constitute the main contribution of this thesis. We describe in considerable detail how four key matrix polynomial problems have successfully been approached through Sylvester matrices and interpolation:

- Section 4.3: Rank evaluation [P4].
- Section 4.4: Triangularization [P1, P5].
- Section 4.5: Continuous-time symmetric bilateral equation [P3].
- Section 4.6: Discrete-time symmetric bilateral equation [P6, P7].

In each section we illustrate our Sylvester matrix and interpolation algorithms on simple numerical examples. In addition to that, we provide throughout the manuscript some intensive computational results that illustrate the power of our approach with comparison to already existing methods. Numerical experiments emphasize that our resolution methods have shown the most promise so far.

4.1 Relevance of polynomial matrices in control

First, we set the ground of the polynomial approach to systems control by proposing a comprehensive classification of matrix polynomial equations stemming from control problems. Some of these equations are also encountered in signal processing and filtering. We do not only confine our attention to basic control tasks such as dead-beat controller design or LQG regulation but also mention more involved problems such as $\mathcal{H}_\infty$ optimization, MIMO model matching or non-interacting control, to mention just a few. We essentially distinguish between linear and quadratic matrix polynomial equations because it seems that there is no practical need for more general forms, at least in the realm of systems control. In this thesis, we deliberately restrict ourselves to linear equations, that is to say, to matrix equations where unknown polynomial coefficients enter linearly. Some results have recently been published on general quadratic matrix polynomial equations [3], but this is mostly an emerging domain of investigation.

By proper manipulations, we will show how key steps in the resolution of the most part of control problems are embodied into the resolution of the general linear matrix polynomial equation

$$A(z)X(z) = B(z). \tag{4.1}$$

This general form allows for accurate definitions of homogeneous solution, polynomial null-space and minimum degree solution. Most notably, it unifies all the concepts previously defined on particular instances of equation (4.1). Note however that such an unification was out of the scope of the present work.

We also mention a different kind of linear equation that cannot be reduced to the above form. It arises when solving particular quadratic matrix polynomial equations (e.g. J-spectral factorization) via linearization schemes and reads

$$A^\star(z)X(z) + X^\star(z)A(z) = B(z) \tag{4.2}$$

where the star denotes the transpose conjugate. At first glance, unified notions of homogeneous solution, polynomial null-space and minimum degree solution do not seem to carry out easily for equation (4.2).

4.1.1 Operations on polynomial matrices

When facing control problems through the polynomial approach, several preliminary steps have usually to be performed. They aim at transforming the polynomial matrix describing the system under study into a standard or canonical form. This category includes the following operations.

- State-space realization to coprime matrix fraction description (MFD) [31, p.123].
- Non-coprime left MFD to coprime right MFD or vice-versa [22, §6.3.1] [11, p.607].

- Reduction of a polynomial matrix into row-reduced or column-reduced form [22, §6.3.2] [11, p.599].
- Extraction of a greatest common divisor of two (or more) polynomial matrices [22, §6.3.1] [11, p.592].
- Transformation of a polynomial matrix into triangular form [22, §6.3.1].

All these operations correspond to solving an homogeneous equation (4.1) with zero right hand-side matrix. Roughly speaking, they are equivalent to the extraction of the polynomial null-space of $A(z)$.

4.1.2 Unilateral matrix polynomial equations

Once the input polynomial matrices are in the correct form, some control problems amount to solving a unilateral equation of the type

$$\hat{A}(z)\hat{X}(z) + \hat{B}(z)\hat{Y}(z) = \hat{C}(z) \tag{4.3}$$

where $\hat{A}(z)$, $\hat{B}(z)$, $\hat{C}(z)$ are given and $\hat{X}(z)$, $\hat{Y}(z)$ are to be found. This equation is referred to as a polynomial Diophantine equation and arises in the following control problems.

- Closed-loop dynamics assignment of a multivariable linear system, by state or output feedback [31, p.123, p.232].
- Dead-beat regulation and tracking [28], considered as a special case of dynamic assignment where $\hat{C}(z)$ is the identity matrix [31, §3.2].
- Extraction of a factor by polynomial division [22, §6.3.2] [11, p.604].

It must be noted that usually, one has some degree requirements on the solution $\hat{X}(z)$, $\hat{Y}(z)$ to equation (4.3). For instance, static state feedback, static state observer or static output injection can be devised when looking for a constant solution $\hat{X}(z)$, $\hat{Y}(z)$, see[32] [31, §3.1, §4.1]. Dead-beat regulation and tracking is generated when seeking a solution of least degree in $\hat{X}(z)$, see [28]. A parameterization of solutions to (4.3) of prespecified degrees may also prove useful [33]. Note that, upon writing

$$A(z) = \begin{bmatrix} \hat{A}(z) & \hat{B}(z) \end{bmatrix}, \quad X(z) = \begin{bmatrix} \hat{X}(z) \\ \hat{Y}(z) \end{bmatrix}, \quad B(z) = \hat{C}(z)$$

then Diophantine equation (4.3) is nothing but a special form of linear matrix polynomial equation (4.1).

4.1.3 Bilateral matrix polynomial equations

Some other control problems, usually more involved than those considered above, yield two coupled bilateral polynomial equations

$$\begin{aligned} \hat{A}(z)\hat{X}(z) + \hat{Z}(z)\hat{B}(z) &= \hat{C}(z) \\ \hat{A}(z)\hat{Y}(z) + \hat{Z}(z)\hat{D}(z) &= \hat{E}(z) \end{aligned} \tag{4.4}$$

where $\hat{A}(z)$, $\hat{B}(z)$, $\hat{C}(z)$, $\hat{D}(z)$, $\hat{E}(z)$ are given and three unknowns $\hat{X}(z)$, $\hat{Y}(z)$, $\hat{Z}(z)$ must be determined. The above equations arise in the following control problems.

- Linear Quadratic Gaussian (LQG) regulation or prediction problems with state or output feedback [31, §3.3, §4.3, §5.3].
- Stationary LQG control problem for descriptor systems [29].
- Standard $\mathcal{H}_2$ control problem [18].
- LQG regulation based on output feedback and disturbance measurement feed-forward. Then two independent couples of polynomial equations (4.4) are derived which yield the feedback and the feed-forward parts of the regulator [44, 17].

Similarly to the Diophantine equation (4.3), degree constraints must be imposed on the solutions $\hat{X}(z)$, $\hat{Y}(z)$, $\hat{Z}(z)$ depending on the nature of the control design. For instance, static feedback implies that $\hat{X}(z)$ and $\hat{Y}(z)$ must be constant. Note also that there is usually some degree constraints on the solution $\hat{Z}(z)$, see [21]. Finally, it is important to underline the fact that coefficients in equations (4.4) are obtained by solving a spectral factorization problem, a special type of quadratic matrix polynomial equation.

Upon denoting

$$A(z) = \begin{bmatrix} \mathbf{I}\otimes\hat{A}(z) & \mathbf{0} & \hat{B}'(z)\otimes\mathbf{I} \\ \mathbf{0} & \mathbf{I}\otimes\hat{A}(z) & \hat{D}'(z)\otimes\mathbf{I} \end{bmatrix},\ X(z) = \begin{bmatrix} \mathrm{vec}\hat{X}(z) \\ \mathrm{vec}\hat{Y}(z) \\ \mathrm{vec}\hat{Z}(z) \end{bmatrix},\ B(z) = \begin{bmatrix} \mathrm{vec}\hat{C}(z) \\ \mathrm{vec}\hat{E}(z) \end{bmatrix}$$

and using standard properties of the Kronecker product (see Section 3.3) it can be checked that coupled bilateral equations (4.4) may be considered as a special form of linear matrix polynomial equation (4.1).

4.1.4 General linear matrix polynomial equations

More general types of linear matrix polynomial equations are integral part of controller synthesis for linear systems. They stem from the general equation

$$\hat{A}(z)\hat{X}(z)\hat{B}(z) + \hat{C}(z)\hat{Y}(z)\hat{D}(z) = \hat{E}(z) \tag{4.5}$$

where $\hat{A}(z)$, $\hat{B}(z)$, $\hat{C}(z)$, $\hat{D}(z)$, $\hat{E}(z)$ are given rational matrices and $\hat{X}(z)$, $\hat{Y}(z)$ are rational matrices to be found. Note that, by appropriate manipulations including least common multiple extraction and the use of the Kronecker product, rational matrix equation (4.5) can always be written as a linear matrix polynomial equation of the type (4.1). In [42], several instances of equation (4.5) are enumerated.

- $\hat{A}(z)\hat{X}(z)\hat{B}(z) = \hat{C}(z)$, in disturbance decoupling problems [42, p.79], non-interacting control problems [42, p.163] and model-matching [42, p.97].
- $\hat{A}(z)\hat{X}(z) + \hat{Y}(z)\hat{C}(z) = \hat{D}(z)$ in regulator problems (asymptotic tracking of a constant signal or asymptotic rejection of a disturbance signal) [42, p.102] and non-interacting control problems [42, p.158]. This equation is known as the skew primeness equation.
- General form (4.5) in regulator problems with measurement feedback [42, p.108].

4.1.5 Symmetric polynomial matrix equations

Finally, in control problems there is another type of linear matrix polynomial equation that, surprisingly and in contrast to the equations described in the previous sections, cannot be reduced to equation (4.1). It reads

$$A^*(z)X(z) + X^*(z)A(z) = B(z) \tag{4.6}$$

where the star denotes the conjugate transpose, i.e. $A^*(s) = A^T(-s)$ in continuous-time and $A^*(d) = A^T(d^{-1})$ in discrete-time.

Polynomial matrix spectral factorization, a standard technique applied in many $\mathcal{H}_2$ control and filtering problems –such as LQ optimal regulation, Kalman filtering, minimum variance LQG control – is traditionally handled by a Newton-Raphson iterative scheme consisting of solving symmetric polynomial matrix equation (4.6) in every step [19]. Next to its control application, equation (4.6) is often encountered in the field of signal processing. There it is namely used in the polynomial approach to computation of an ARMA covariance function [45].

4.2 Existing numerical methods for polynomial matrices

In this section, we spell out already existing numerical methods that handle polynomial matrices and linear matrix polynomial equations. They may be cast into several categories as follows.

- Elementary polynomial operations, an extension to polynomials of the standard Euclidean division algorithm [27] [22, §6.3], [10, §2.3].
- Polynomial reduction methods, a sophisticated version of elementary polynomial operations [19, 21].
- State-space realization methods, where state-space techniques are used as a bypass to solve polynomial equations [38]. Some of these methods are mixed, for they also feature elementary polynomial operations [12, 54].
- Matrix pencil algorithms, relying upon Kronecker invariants [5, 6, 24].
- Symbolic methods, using exact computations over rational numbers [47, 51].
- Sylvester matrices, based upon equating coefficients of the powers of the indeterminate [22, 11, 46].
- Interpolation, based upon evaluating values taken by polynomial matrices at given points of the complex plane [2].

The advantages and shortcomings of each method are investigated in the sequel, with the objective of showing how unsatisfactory prevailing techniques can be, as far as computational efficiency and numerical reliability are concerned. It clearly points out that there is still a need for efficient and reliable algorithms for performing tasks related to polynomial matrices.

4.2.1 Elementary polynomial operations

Many of the results on elementary polynomial operations were first obtained for matrices with integer entries. It was only realized much later that they also held for entries that are polynomials and more generally for entries drawn from any principal ideal domain.

The elementary column (or row) operations for polynomial matrices are described as follows.

- Interchange of any two columns (or rows).
- Addition to any column (row) of a polynomial multiple of any other column (row).
- Scaling any column (row) by any nonzero real or complex number.

These elementary operations can be represented by elementary unimodular matrices, that is to say, polynomial matrices whose determinant is not a function of the indeterminate.

Elementary operations are undoubtedly the most widespread method dealing with polynomials and polynomial matrices. This is most likely due to historical reasons and also because it remains the simplest and more intuitive way of solving polynomial equations.

By practical experiments, it has often been pointed out that elementary polynomial operations are prone to numerical instability. In other words, they may sometimes return spurious results and behave erratically. The reasons behind this are not well-known and are still subject to investigation. An error analysis should be performed as in [15, 16] that would theoretically illustrate the potential instability of these methods.

On the positive side, elementary polynomial operations are usually fairly accurate when successful and require a low amount of memory [30, 25]

4.2.2 Polynomial reduction methods

Polynomial reduction methods are a sophisticated version of the Euclidean division algorithm. The basic idea is as follows: by an appropriate polynomial substitution, the original polynomial equation is transformed into an equation involving polynomials of lower degrees. This scheme is successively applied until one eventually obtains an equation over constant numbers. The constant equation is readily solved, and by proper backwards substitutions, polynomial solutions to the original polynomial equations are recovered.

The approach, well explained and illustrated in [19] and [21], is particularly elegant and requires a low amount of memory. On the other hand, it is closely related to the elementary polynomial operations approach and therefore inherits poor numerical properties.

4.2.3 State-space techniques

By establishing the relationship between the behavior of a dynamical system [52] and the null-space of a polynomial matrix, Kwakernaak [38] shows that various problems related to polynomial matrices can be dealt with in the state-space setting. This is particularly interesting because numerical properties of state-space algorithms are usually rather good.

However, potential numerical difficulties may be encountered when passing from the polynomial representation to the state-space setting and vice-versa. Research is underway to evaluate the relative performance of this kind of algorithms.

Some other algorithms are based upon mixed elementary polynomial operations and state-space realization methods, hence inheriting the numerical properties of both techniques [12, 54].

4.2.4 Matrix pencil algorithms

Matrix pencil algorithms seem to be promising, but have only been developed for very specific problems as yet. They rely upon the computation of the Kronecker form of matrix pencils, for which numerically reliable algorithms were developed [48]. Pencil algorithms are used in [5] for deriving a coprime factorization of a transfer matrix, in [6] for computing the inverse and the null-space of a polynomial matrix and in [24] for extracting the greatest common divisor of a set of polynomials. However, no unified approach seems to be available for the moment. This domain is currently subject to investigation, as illustrated by the recent theoretical work [43].

4.2.5 Symbolic methods

Symbolic methods for manipulating polynomial matrices turn out to be of no practical use, except for problems of very little dimensions. This is due to intermediate expression swell, a combinatorial explosion of the amount of memory that is required to perform symbolic computations. However, several recent works [14, 47, 51] tend to illustrate that progress is underway for overcoming some of the fundamental limitations of this approach.

4.2.6 Sylvester matrices

Sylvester matrices have been used satisfactorily for solving several instances of matrix polynomial equations [46], but no general results have been carried out so far. On the positive side, it can be underlined that the formulation of polynomial equations in terms of Sylvester matrices is straightforward. It results in a linear system of equations, for which several resolution methods have been developed [15]. On the negative side, the method usually requires a substantial amount of memory and this has precluded its systematic use some fifteen years ago. One of the objective of our work is to illustrate that this is not true anymore. In the sequel, we will show by practical experiments that the availability of powerful computational tools has turned this technique into one of the most efficient.

4.2.7 Interpolation

The above comments also apply to the interpolation approach, which has been shown to be closely related to the Sylvester matrix approach. However, an extra difficulty is introduced by the choice of the interpolation points. Indeed, it is well-known that this choice strongly influences the conditioning of the Vandermonde matrix [16, §21.1]. This aspect will be emphasized throughout the manuscript.

4.3 Rank evaluation

4.3.1 Introduction

Rank evaluation over the ring of polynomials is a fundamental component of a large number of algorithms dealing with polynomial matrices. It is a necessary preliminary step when developing numerical routines in the scope of the polynomial approach to linear systems analysis and design. Rank evaluation may also prove useful when testing relative primeness of polynomial matrices, see Lemma 6.3-6 in [22]. In spite of this, already existing numerical methods for polynomial matrix rank evaluation cannot be considered as reliable. Indeed, they are based on elementary polynomial operations that are known to suffer from bad numerical properties.

Accordingly, our contribution is to propose two simple and new algorithms for polynomial matrix rank evaluation. They only hinge on real matrix rank evaluation, hence relying upon powerful and well tried numerical tools such as the singular value decomposition [15]. The first algorithm is based on interpolation whereas the second algorithm uses Sylvester matrices.

Throughout the presentation of these algorithms, closely related issues such as polynomial null-space extraction and existence of solutions to matrix polynomial equations arising in control problems are furthermore touched on.

4.3.2 Definitions of the rank

There are several definitions of the rank of a polynomial matrix. Hopefully all of them are equivalent. Hereafter is a list which is far from being exhaustive but that covers major statements found in control theory textbooks [10, 11, 22, 27, 31].

Definition 4.1 *The* rank *(or* normal rank *or* determinantal rank*) of a polynomial matrix $A(z)$ is*

1. *the number of linearly independent columns (resp. rows) of A in the polynomial sense*
2. *the number of non-zero columns (resp. rows) of the column (resp. row) Hermite form of A*
3. *the number of invariant polynomials of the Smith form of A*
4. *the largest order of minors of A that are not identically zero*
5. *the dimension of the module generated by the columns (resp. rows) of A*

6. *the difference between the number of columns (resp. rows) of A and the dimension of the polynomial right (resp. left) null-space of A*

7. *the number*

$$\max_{\lambda \in \mathbb{C}} \operatorname{rank} A(\lambda)$$

4.3.3 Interpolation algorithm

The first algorithm for polynomial matrix rank evaluation is based on interpolation. The indeterminate z is assigned a set of complex values $z_0, z_1, z_2, \ldots$ and ranks of constant matrices $A(z_i)$ are successively evaluated. Generally speaking, interpolation is based on the following intuitive reasoning: a property or value of a polynomial matrix $A(z)$ is inferred from the same property or value assessed on a set of constant interpolated matrices $A(z_0), A(z_1), A(z_2), \ldots$ The sequel aims at demonstrating that this approach can also straightforwardly be employed for rank evaluation.

Recall that one standard definition of the rank of a polynomial matrix is

$$r = \operatorname{rank} A(z) = \max_{\lambda \in \mathbb{C}} \operatorname{rank} A(\lambda) \tag{4.7}$$

The following concept of a zero plays a key role once combined with relation (4.7).

Definition 4.2 *The invariant zeros of the determinant of the minors of polynomial matrix $A(z)$ are the complex values z_i at which A drops below its normal rank, i.e.*

$$\operatorname{rank} A(z_i) < r. \tag{4.8}$$

They are called the zeros *of A.*

Fact 4.1 *The number of zeros of a polynomial matrix $A \in \mathbb{C}^{m \times n}[z]$ is always less than or equal to the degree of the determinant of the largest minor of A. In other words, there are only up to*

$$k = \min\left(\sum_{i=1}^{m} \deg \operatorname{row}_i A, \sum_{j=1}^{n} \deg \operatorname{col}_j A\right) \tag{4.9}$$

complex values z_i such that (4.8) holds.

Finally, a topological property of the zeros of a polynomial – see §0.16 in [55] – will be the last component required to design the interpolation algorithm.

Fact 4.2 *If $z_0 \in \mathbb{C}$ is a zero of a polynomial $p(z) \in \mathbb{C}[z]$ and if p vanishes identically in some neighborhood of z_0, then p vanishes on $\mathbb{C}$.*

Clearly, it is always possible to pick up a set of $k+1$ real values $z_0, z_1, \ldots, z_k$ that are sufficiently far from each other. According to Definition 4.2 and Properties 4.1 and 4.2, at least one element z_i of this set *is not a zero* of A, hence

$$\text{rank}A(z_i) = \max_{j=0,\ldots,k} \text{rank}A(z_j).$$

The exposition of the first algorithm is now in order.

Algorithm RankInt Given a polynomial matrix $A(z) \in \mathbb{C}^{m\times n}[z]$, the following algorithm evaluates the rank of A to working precision.

Step 1 Select a random number $z_0 \in \mathbb{C}$. Let $i = 0$ and $r = 0$.
Step 2 Let $r = \max(r, \text{rank}A(z_i))$. If $r = \min(m, n)$ then return r.
Step 3 Let $i = i + 1$. If $i > k$, where k is given in (4.9), then return r.
Step 4 Select a random number $z_i \in \mathbb{C}$ different enough from previous numbers $z_0, z_1, \ldots, z_{i-1}$. More specifically, select z_i such that for a sufficiently large real ε, the inequality $||z_i| - |z_j|| \geq \varepsilon |z_i||z_j|$ holds for any $j = 0, 1, \ldots, i-1$. Then go to Step 2.

It may be pointed out that if A has full rank then usually Algorithm RankInt stops *almost immediately* in Step 2 since it is very unlikely that a random number z_0 be a zero of A. Conversely, if A is rank deficient then $k+1$ constant matrix rank evaluations will be needed.

4.3.4 Sylvester matrix algorithm

A second rank evaluation technique is now spelled out. It originates from the concept of Sylvester matrix and is tightly connected to the notion of the null-space of a polynomial matrix. As a result, a necessary condition of existence of solutions to general linear matrix polynomial equations is exposed hereafter.

Rank evaluation and polynomial null-space

Given a polynomial matrix $A(z) \in \mathbb{C}^{m\times n}[z]$, suppose that all q linearly independent polynomial vectors $k_i(z) \in \mathbb{C}^n[z]$ such that $A(z)k_1(z) = A(z)k_2(z) = \ldots = A(z)k_q(z) = \mathbf{0}_{m\times 1}$ are collected to form a polynomial matrix

$$K(s) = [k_1(z) \quad k_2(z) \ \ldots \ k_q(z)].$$

Matrix K is clearly a representation of the polynomial right null-space of A and verifies

$$A(z)K(z) = \mathbf{0}_{m\times q}. \tag{4.10}$$

From Definition 1.6, it holds

$$\text{rank}A = n - q. \tag{4.11}$$

Now write A and K in terms of increasing powers of the indeterminate z

$$\begin{array}{rcl} A(z) & = & A_0 + zA_1 + \ldots + z^{d_A} A_{d_A} \\ K(z) & = & K_0 + zK_1 + \ldots + z^{d_K} K_{d_K}. \end{array}$$

By equating the powers of z in matrix polynomial equation (4.10), one can build an equivalent linear system of equations

$$\underbrace{\begin{bmatrix} A_0 & & & \mathbf{0} \\ A_1 & A_0 & & \\ \vdots & A_1 & \ddots & \\ A_{d_A} & \vdots & & A_0 \\ & A_{d_A} & & A_1 \\ & & \ddots & \vdots \\ \mathbf{0} & & & A_{d_A} \end{bmatrix}}_{\langle A \rangle_{d_K}} \underbrace{\begin{bmatrix} K_0 \\ K_1 \\ \vdots \\ K_{d_K} \end{bmatrix}}_{\overline{K}} = \mathbf{0} \tag{4.12}$$

involving $\langle A \rangle_{d_K}$, the Sylvester matrix of $A(z)$ of order d_K. Since matrix polynomial equation (4.10) and linear system (4.12) are equivalent, extracting polynomial null-space K amounts to computing the null space of constant matrix $\langle A \rangle_{d_K}$, the only issue remaining the estimation of degree d_K. This has motivated the following lemma which provides an upper bound on d_K and therefore on the size of the Sylvester matrix.

Lemma 4.1 *Given a polynomial matrix $A(z) \in \mathbb{C}^{m \times n}[z]$, let*

$$d_R^* = \sum_{i=1}^{n} \deg \operatorname{col}_i A - \min_{i=1,\ldots,n} \deg \operatorname{col}_i A. \tag{4.13}$$

Then the polynomial right null-space of A is generated by the columns of a polynomial matrix of degree less than or equal to d_R^*.

Proof Let $\operatorname{rank} A = n - q$ and U be a unimodular matrix of degree d_U that transforms A into column Hermite form [P5], i.e.

$$\begin{array}{rcl} A(z)U(z) & = & A(z)[V(z) \quad K(z)] \\ & = & [A(z)V(z) \quad \mathbf{0}_{m \times q}]. \end{array}$$

The $n \times q$ polynomial matrix K is a polynomial right null-space of A and obviously

$$d_K \leq d_U. \tag{4.14}$$

Now suppose that P is a constant permutation matrix such that

$$P \begin{bmatrix} A(z) \\ \mathbf{I}_n \end{bmatrix} U(z) = \begin{bmatrix} M(z) \\ \star \end{bmatrix} U(z) = \begin{bmatrix} N(z) \\ \star \end{bmatrix}$$

where a star denotes an irrelevant block, M and N are non-singular and N is lower triangular. Since U is unimodular,

$$\deg \det N = \deg \det M = \sum_{i=1}^{n} \deg \mathrm{col}_i N \tag{4.15}$$

Let M_{adj} stand for the adjoint polynomial matrix of M, whence

$$U(z) \det M(z) = M_{\mathrm{adj}}(z) N(z). \tag{4.16}$$

From (4.16) and (4.15), it follows that

$$\begin{aligned} d_U &\leq \deg M_{\mathrm{adj}} + \deg N - \deg \det M \\ &\leq \deg M_{\mathrm{adj}} + \max_{i=1,\dots,n} \deg \mathrm{col}_i N - \textstyle\sum_{i=1}^{n} \deg \mathrm{col}_i N. \end{aligned} \tag{4.17}$$

Since entries of M_{adj} are minors of A of order $n-1$, we have

$$\deg M_{\mathrm{adj}} \leq d_R^* \tag{4.18}$$

where d_R^* is given in (4.13). The claim then follows from (4.18), (4.17) and (4.14). □

In [7] it is stated that the bound d_R^* is sharp. As a direct consequence of the above lemma, it is sufficient to extract the right null-space of constant matrix $\langle A \rangle_{d_R^*}$ in order to derive a basis for the polynomial right null-space of A.

A similar result for the polynomial left null-space of A is captured by the following corollary to Lemma 4.1.

Corollary 4.1 *Given a polynomial matrix $A(z) \in \mathbb{C}^{m \times n}[z]$, let*

$$d_L^* = \sum_{i=1}^{m} \deg \mathrm{row}_i A - \min_{i=1,\dots,m} \deg \mathrm{row}_i A. \tag{4.19}$$

Then the polynomial left null-space of A is generated by the rows of a polynomial matrix of degree less than or equal to d_L^*.

For notational ease, assume henceforth that $\langle A \rangle_d = \mathbf{0}$ if $d < 0$. The next theorem summarizes the implications of Lemma 4.1 and Corollary 4.1 for rank evaluation and simultaneously provides a second algorithm.

Theorem 4.1 (Algorithm RankSyl) *Given a polynomial matrix $A(z) \in \mathbb{C}^{m \times n}[z]$,*

$$\mathrm{rank} A = \min \left\{ \mathrm{rank} \langle A \rangle_{d^*} - \mathrm{rank} \langle A \rangle_{d^*-1}, \mathrm{rank} \langle A' \rangle_{d^*} - \mathrm{rank} \langle A' \rangle_{d^*-1} \right\} \tag{4.20}$$

where

$$d^* = \min(d_R^*, d_L^*) \tag{4.21}$$

and d_R^*, d_L^* are respectively given in (4.13) and (4.19).

Proof First consider the subspace generated by the columns of Sylvester matrix $\langle A\rangle_d$ for a given order d. Let q_i be the number of linearly dependent columns in the $(i+1)$th block column of $\langle A\rangle_d$. Because of the structure of the Sylvester matrix, q_i increases monotonically with i – see §G.5 in [11]. Furthermore, Lemma 4.1 provides an upper bound $q_{d_R^*}$ such that

$$\begin{array}{l} 0 \leq q_0 \leq q_1 \leq \ldots \leq q_{d_R^*} \\ q_{d_R^*} = q_{d_R^*+1} = q \end{array} \tag{4.22}$$

where q is the dimension of the polynomial right null-space of A. Hence for any $d \geq 0$,

$$\begin{aligned} \operatorname{rank}\langle A\rangle_d &= (n-q_0)+\cdots+(n-q_d) \\ &= (d+1)n - \sum_{i=0}^{d} q_i. \end{aligned} \tag{4.23}$$

From relations (4.11), (4.22) and (4.23), one can check that

$$\begin{aligned} \operatorname{rank} A &= \operatorname{rank}\langle A\rangle_{d_R^*} - \operatorname{rank}\langle A\rangle_{d_R^*-1} \\ &= n - q. \end{aligned} \tag{4.24}$$

Similarly, consider now the subspace generated by the rows of Sylvester matrix $\langle A\rangle_d$. Using Corollary 4.1 and the same reasoning as above, it holds

$$\operatorname{rank} A = \operatorname{rank}\langle A\rangle_{d_L^*} - \operatorname{rank}\langle A\rangle_{d_L^*-1}. \tag{4.25}$$

Relations (4.20) and (4.21) then readily follow from (4.24) and (4.25). □

Similarly, it is possible to decide whether a polynomial matrix has full column rank or not, and by extension, whether a polynomial matrix is singular or not.

Corollary 4.2 *A polynomial matrix $A(z) \in \mathbb{C}^{m\times n}[z]$ has full column rank if and only if $\langle A\rangle_{d^*}$ has full column rank, where d^* is given in (4.21).*

Proof If A has full column rank then according to Theorem 4.1, $\operatorname{rank}\langle A\rangle_{d^*} - \operatorname{rank}\langle A\rangle_{d^*-1} = \operatorname{rank} A = n - q_{d^*} = n$. Since $q_{d^*} = 0$, $\langle A\rangle_{d^*}$ has full column rank. Conversely, if $\langle A\rangle_{d^*}$ has full column rank then there is no solution to (4.12) when $d_K = d^*$ and as shown by Lemma 4.1, the right null-space of A is zero. □

Corollary 4.2 can be considered as a computationally tractable extension of Lemma G-2 in [11] where the rank condition was only established on an infinite set of Sylvester matrices of increasing orders $\langle A\rangle_0$, $\langle A\rangle_1$, $\langle A\rangle_2, \ldots$.

Linear matrix polynomial equations

As pointed out previously, the equation

$$A(z)X(z) = B(z) \tag{4.26}$$

can be considered as the most general type of linear matrix polynomial equation. The results developed above will now be applied to derive a numerically tractable necessary condition of existence of a solution to (4.26).

Lemma 4.2 *Given two polynomial matrices $A(z) \in \mathbb{C}^{m\times n}[z]$ and $B(s) \in \mathbb{C}^{m\times p}[z]$, let*

$$d_j^\dagger = \sum_{i=1}^{n} \deg \mathrm{col}_i A + \deg \mathrm{col}_j B - \min\left(\min_{i=1,\ldots,n} \deg \mathrm{col}_i A, \quad \deg \mathrm{col}_j B\right) \tag{4.27}$$

and

$$d^\dagger = \max_{j=1,\ldots,p} d_j^\dagger. \tag{4.28}$$

Then if the Sylvester matrix

$$< [A(z) \;\; B(z)] >_{d^\dagger} \tag{4.29}$$

has full rank there is no polynomial solution $X(z)$ to equation (4.26).

Proof First write (4.26) column-wise :

$$A(z)[x_1(z) \; \ldots \; x_p(z)] = [b_1(z) \; \ldots \; b_p(z)].$$

If (4.26) holds then for any $j = 1, 2, \ldots, p$,

$$[A(z) \quad b_j(z)] \begin{bmatrix} x_j(z) \\ -1 \end{bmatrix} = \mathbf{0}_{m\times 1}.$$

From Lemma 4.1, the degree of the polynomial right null-space of matrix $[A(z) \; b_j(z)]$ is bounded from above by $d_j^\dagger$. Since this holds for any $j = 1, 2, \ldots, p$, a necessary condition for a polynomial solution X to (4.26) to exist is that a null-space of degree less than or equal to $d^\dagger$ exists for matrix $[A(z) \; B(z)]$. The rank condition on constant matrix (4.29) then directly proceeds from the Sylvester matrix formulation. □

The next example will finally aim to reveal that Lemma 4.2 can shed light on classical resolution problems.

Example 4.1 *Suppose that one is seeking a polynomial solution X to the polynomial matrix equation*

$$\begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix} X(z) = \begin{bmatrix} z \\ 1 \end{bmatrix}. \tag{4.30}$$

A first glance to respective degrees of polynomial matrices appearing at both sides of the above equation could suggest that it is sufficient to look for a solution $X(z)$ of degree 1. However Lemma 4.2 states that it is necessary to expect solutions X of degree up to $d^\dagger = 2$.

Sylvester matrix (4.29) then reads

$$\left[\begin{array}{ccc|ccc|ccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}\right]$$

and does not have full rank. Thus, a solution of degree 2 *may exist. Indeed, using standard resolution tools [P10] it can be checked that*

$$X(z) = \begin{bmatrix} z \\ 1 - z^2 \end{bmatrix}$$

is (the unique) solution to equation (4.30).

4.4 Triangularization

4.4.1 Introduction

A common theme when performing computations involving polynomials is to transform a given polynomial matrix into a simpler equivalent form while preserving key properties of the original matrix. The triangularization over the ring of polynomials is a fundamental construction in this regard.

Reduction of polynomial matrices into triangular form has crucial relevance in linear system theory. Applications include solving matrix polynomial equations such as the Diophantine equation, extracting the greatest common divisor of polynomial matrices [10, 11, 22], computing the determinant of a polynomial matrix [P9], as well as obtaining a triangular basis for modules over polynomial domains [10].

The first results were obtained by Hermite for matrices with integer entries and it was only realized later that they also held for entries that are polynomials. This led to the classical triangularization algorithm exposed in [10, 11, 22], a simple variation of Gaussian elimination with the extended Euclidean algorithm replacing division. It is well-known that this procedure results impractical for its bad numerical behavior.

The idea of converting triangularization over the ring of polynomials to that of solving linear systems over the field of reals appears to have first been used by Kaltofen *et al.* [23]. The goal of the authors was to establish a parallel complexity result and Storjohann [47, Chapter 4] reexamined their approach to obtain sequential results. The key point was to note that one is led to a linear system that does not involve unknown coefficients of the triangular form, but only coefficients of the transformation matrix. Parallel and sequential complexity bounds are also proposed in [51], where the triangular form is derived from the Popov form, also widely used in linear system control theory.

In a similar vein, Labhalla *et al.* [39] recently developed a particularly elegant linear system method based on the theory of sub-resultants and Sylvester matrices. Their procedure is actually a specialization to triangularization of the greatest common divisor extraction algorithm described by Bitmead *et al.* in the early paper [8].

Although it has not rigorously been proved, one can reasonably argue that the approach described in [8, 39] features numerical shortcomings that are similar to those of the standard Euclidean division algorithm. Indeed, the algorithm found in these references is designed to preserve the shift-invariant Toeplitz structure of the Sylvester matrix and is actually a modified form of Gaussian elimination with only partial pivoting. As a result, complete numerical stability cannot be assured. In [8] it is stated that this seems to be a fundamental limitation since one is naturally constrained to use unimodular rather than orthogonal transformations when performing polynomial reductions.

In a more general context, the interpolation approach, clearly exposed in [2] by Antsaklis and Gao has been shown to supply solutions to the most part of control problems involving polynomial matrices. Interpolation techniques also provide, in an elegant and flexible way, alternative interpretations of well-known linear system theory results. It should thus seem surprising that, no results are available so far concerning an interpolation approach to triangularization of polynomial matrices.

Based on the above considerations, our objective is twofold. At the one hand, we fulfill the need for polynomial matrix triangularization algorithms that are numerically reliable, with contrast to already published methods. At the other hand, we aim at extending the domain of application of interpolation by showing that this useful technique also applies to triangularization. More specifically, two almost equivalent numerical procedures are proposed. The first one is based on Sylvester matrices, whereas the second one exclusively relies on interpolation theory.

Our developments are organized as follows. Classical facts are first recalled on the reduction into triangular form of matrices over the field of polynomial fractions and over the ring of polynomials. A useful lemma is exposed that provides an upper bound on the degree of the reduction matrix. Equivalence is then established between triangularization over the field of polynomial fractions and triangularization over the fields of real numbers. Sylvester matrices and interpolation theory are used for this purpose. Both approaches are shown to be closely related. Several aspects concerning interpolation are dealt with, such as the choice and the number of interpolation points. Two numerically stable algorithms are described for triangularization over the field of polynomial fractions. They can alternatively be used for polynomial rank evaluation and polynomial null-space extraction. Some algebraic complements about degree minimization over ideals are then exposed. They will be instrumental to the derivation of the triangular form over the ring of polynomials, using equivalently Sylvester matrices or interpolation theory. Two algorithms are then proposed that only rely on numerically stable routines. Our techniques are then illustrated on a very simple example. Finally, comparative numerical results are exposed. They aim at illustrating advantages of our approach over other triangularization methods.

4.4.2 Preliminaries

In this section, the notion of a shape of a polynomial matrix is defined. Some standard facts on reduction of polynomial matrices into triangular forms are recalled. Distinction is made between triangular forms over field $\mathbb{R}(z)$ and triangular forms over ring $\mathbb{R}[z]$. Finally, a useful upper bound is derived on the degree of the polynomial matrix that is used to triangularize a polynomial matrix.

Definition 4.3 *The shape Σ of a full column rank n-by-m matrix A is the series $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ of positive integers $\sigma_i = 1, 2, \ldots, n$ such that $\mathrm{ent}_{\sigma_i,i}A$ is the uppermost non-zero entry of $\mathrm{col}_i A$, called leading entry.*

Definition 4.4 *A shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ whose components are strictly increasing, i.e. $\sigma_1 < \ldots < \sigma_m$, is called a lower left triangular shape. A matrix with lower left triangular shape is in lower left triangular form.*

Definition 4.5 *Elementary column operations for polynomial matrices are defined over $\mathbb{R}(z)$ (respectively over $\mathbb{R}[z]$) as*

- *Interchange of any two columns.*
- *Addition to any column of a polynomial multiple of any other column.*
- *Scaling any column by any nonzero element of $\mathbb{R}[z]$ (resp. $\mathbb{R}$).*

These elementary operations can be represented by post-multiplication by a polynomial matrix invertible over $\mathbb{R}(z)$, or non-singular (resp. invertible over $\mathbb{R}[z]$, or unimodular).

Theorem 4.2 *Any polynomial matrix $A(z)$ can be transformed into lower left triangular form $T(z)$ by application of elementary column operations over $\mathbb{R}(z)$ (respectively over $\mathbb{R}[z]$), or equivalently by post-multiplication by a non-singular (resp. unimodular) reduction matrix $U(z)$, i.e.*

$$A(z)U(z) = T(z). \tag{4.31}$$

Proof : The proof is by construction and consists in an algorithm based on polynomial Euclidean division by elementary column operations. See [10, §2.3.4], [11, §G-3] or [22, §6.3.1]. □

Remark 4.1 A polynomial matrix $A(z)$ has generally several triangular forms over $\mathbb{R}(z)$ or $\mathbb{R}[z]$. However, all triangular forms of $A(z)$ have the same triangular shape. Indeed, if $A(z)$ has two triangular forms with different shapes, on can check that it is not possible to obtain one triangular form from the other using elementary column operations only, thus leading to a contradiction.

Remark 4.2 If every leading entry of a lower left triangular form $T(z)$ over $\mathbb{R}[z]$ is monic and of higher degree than any entry at the left of it, then $T(z)$ is in Hermite column form, also called reduced column echelon form of $A(z)$ over $\mathbb{R}[z]$. The Hermite form of a square matrix is unique, see e.g. [11, §G-3] or [22, §6.7.1].

Some important comments are in order.

- Suppose that the identity matrix is appended at the bottom of an arbitrary rank n-by-m polynomial matrix $A^\dagger(z)$ and let $A(z)$ be the resulting full column rank compound matrix. Triangularization of $A(z)$ reads
$$A(z)U(z) = \left[\begin{array}{c} A^\dagger(z) \\ I_m \end{array}\right] U(z) = \left[\begin{array}{c} T^\dagger(z) \\ U(z) \end{array}\right] = T(z). \tag{4.32}$$
In the above equation $T^\dagger(z)$ is a triangular form of $A^\dagger(z)$. Hence a triangular form of an arbitrary rank polynomial matrix and the corresponding reduction matrix have simultaneously been computed through triangularization of a full column rank matrix. Based on this and without loss of generality, in this chapter only full column rank polynomial matrices will be considered.

- Throughout the chapter only lower left triangular forms and elementary column operations are considered. Obviously, dual results also apply to upper right triangular forms and elementary row operations.

- In contrast to the triangular form over $\mathbb{R}[z]$, the more general triangular form over $\mathbb{R}(z)$ is apparently of no direct use in control theory. It is considered in this chapter for two reasons. First, it may indirectly be used for polynomial matrix rank evaluation or polynomial null-space extraction. Second and most importantly, it proves to be a useful intermediate step towards the derivation of the triangular form over $\mathbb{R}[z]$.

- The algorithms presented in this chapter are only designed to compute triangular forms over $\mathbb{R}(z)$ or $\mathbb{R}[z]$, not necessarily the Hermite form. The reason behind this is numerical and will be exposed later on.

Lemma 4.3 *Given a full column rank polynomial matrix $A(z) \in \mathbb{R}^{n\times m}[z]$, let $r_i \in [1,n]$, $i = 1,\ldots,n$ and $c_j \in [1,m]$, $j = 1,\ldots,m$ be two series of integers such that* $\deg \mathrm{row}_{r_1} A(z) \geq \cdots \geq \deg \mathrm{row}_{r_n} A(z)$ *and* $\deg \mathrm{col}_{c_1} A(z) \geq \cdots \geq \deg \mathrm{col}_{c_m} A(z)$*. Define*
$$\hat{d}_U = \min\left\{\sum_{i=1}^{m-1} \deg \mathrm{row}_{r_i} A(z),\ \sum_{j=1}^{m-1} \deg \mathrm{col}_{c_j} A(z)\right\}. \tag{4.33}$$

Then there always exists a reduction matrix $U(z)$ of degree $d_U \leq \hat{d}_U$ such that $T(z) = A(z)U(z)$ is a triangular form of $A(z)$ over $\mathbb{R}[z]$, hence over $\mathbb{R}(z)$.

Proof : The proof is adapted from [39, Proposition 4] and [47, Theorem 6]. Let $U(z)$ be a unimodular matrix such that $T(z) = A(z)U(z)$ is in lower left triangular form, and let $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ be the shape of $T(z)$ from which is defined the non-singular matrix

$$A_\Sigma(z) = \begin{bmatrix} \mathrm{row}_{\sigma_1} A(z) \\ \vdots \\ \mathrm{row}_{\sigma_m} A(z) \end{bmatrix}$$

and the non-singular lower left triangular matrix $T_\Sigma(z) = A_\Sigma(z)U(z)$. We aim to show that $\deg U(z) \leq \hat{d}_U$. Since $U(z)$ is unimodular and $T_\Sigma(z)$ is square triangular, $\deg\det A_\Sigma(z) = \deg\det T_\Sigma(z) = \sum_{i=1}^m \deg \mathrm{col}_i T(z)$. Now let $A_\Sigma^{\mathrm{adj}}(z)$ denote the polynomial adjoint matrix of $A_\Sigma(z)$, so that $\det A_\Sigma(z)U(z) = A_\Sigma^{\mathrm{adj}}(z)T_\Sigma(z)$. The latter equation yields

$$\begin{aligned} d_U = \deg U(z) &\leq \deg A_\Sigma^{\mathrm{adj}}(z) + \deg T_\Sigma(z) - \deg\det A_\Sigma(z) \\ &\leq \deg A_\Sigma^{\mathrm{adj}}(z) + \max_{i=1,\ldots,m} \deg \mathrm{col}_i T(z) - \textstyle\sum_{i=1}^m \deg \mathrm{col}_i T(z) \\ &\leq \deg A_\Sigma^{\mathrm{adj}}(z). \end{aligned}$$

In order to prove the lemma, it is now sufficient to show that $\deg A_\Sigma^{\mathrm{adj}}(z) \leq \hat{d}_U$, where $\hat{d}_U$ is given in (4.33). To see that, note that entries of $A_\Sigma^{\mathrm{adj}}(z)$ are minors of order $m-1$ of $A_\Sigma(z)$. The degree of the minor obtained by removing the i-th row and j-th column of $A_\Sigma(z)$ is less than or equal to $\sum_{k=1}^m \deg \mathrm{row}_k A_\Sigma(z) - \deg \mathrm{row}_i A_\Sigma(z)$ (row-wise) and is also less than or equal to $\sum_{k=1}^m \deg \mathrm{col}_k A_\Sigma(z) - \deg \mathrm{col}_j A_\Sigma(z)$ (column-wise). Upon applying this reasoning to all the entries in $A_\Sigma^{\mathrm{adj}}(z)$, the required bound is obtained. $\square$

4.4.3 Triangular form over $\mathbb{R}(z)$

In this section, equivalence is established between triangularization over $\mathbb{R}(z)$ and over $\mathbb{R}$. For this purpose, we use Sylvester matrices and interpolation. As a result, two algorithms for triangularization over $\mathbb{R}(z)$ are proposed. Their respective numerical properties are discussed at length.

Sylvester matrix approach

Let $\{e_1, e_2, \ldots, e_n\}$ be the canonical basis of the free module $\mathbb{R}^n[z]$. Given a polynomial matrix $A(z) \in \mathbb{R}^{n\times m}[z]$ of degree d_A, let $\mathcal{A} \subset \mathbb{R}^n[z]$ denote the submodule generated by the columns of $A(z)$. Further, let $\mathbb{R}^n[z]_d \subset \mathbb{R}^n[z]$ represent the submodule spanned by all elements of $\mathbb{R}^n[z]$ of degree less than or equal to d and let $\mathcal{A}_d = \mathcal{A} \cap \mathbb{R}^n[z]_d$ be the corresponding submodule of $\mathcal{A}$.

As in Section 3.3, we define P as the orthogonal permutation matrix of dimension $n(d+1)$. The row permuted Sylvester matrix of $A(z)$ of order $d = d_A + d_U$ is denoted by

$$\boldsymbol{R}_d = P\langle A(z)\rangle_{d_U}. \tag{4.34}$$

Finally, let

$$\{\{z^d e_1, \ldots, ze_1, e_1\}, \cdots, \{z^d e_n, \ldots, ze_n, e_n\}\} \tag{4.35}$$

be a generating basis for $\mathbb{R}^n[z]_d$. By construction, for any $i = 1, \ldots, n$ and $j = 0, \ldots, d_U$, the $(i + (d_U - j)m)$-th column of matrix $\boldsymbol{R}_d$ is a real vector representation with respect to basis (4.35) of the polynomial vector $\mathrm{col}_i(z^j A(z)) \in \mathcal{A}_d \subset \mathbb{R}^n[z]_d$. Let $\mathcal{R}_d \subset \mathcal{A}_d$ denote the submodule spanned by all the elements $\mathrm{col}_i(z^j A(z))$ for $i = 1, \ldots, n$ and $j = 0, \ldots, d_U$. By a slight abuse of terminology, we will henceforth say that matrix $\boldsymbol{R}_d$ generates submodule $\mathcal{R}_d$ in basis (4.35).

Lemma 4.4 *Triangularizing over $\mathbb{R}(z)$ a full column rank polynomial matrix $A(z)$ of degree d_A by a reduction matrix $U(z)$ of degree d_U amounts to triangularizing over $\mathbb{R}$ the row permuted Sylvester matrix $\boldsymbol{R}_d$, where $d = d_A + d_U$.*

Proof Triangularization of $A(z) \in \mathbb{R}^{m\times n}[z]$ by $U(z)$ yields matrix polynomial equation (4.31) where $T(z)$ has triangular shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$. Let

$$U(z) = ([z^{d_U} \ \cdots \ z \ \ 1] \otimes \mathbf{I}_m) \underbrace{\begin{bmatrix} U_{d_U} \\ \vdots \\ U_1 \\ U_0 \end{bmatrix}}_{\bar{U}}, \quad T(z) = ([z^{d} \ \cdots \ z \ \ 1] \otimes \mathbf{I}_n) \underbrace{\begin{bmatrix} T_{d} \\ \vdots \\ T_1 \\ T_0 \end{bmatrix}}_{\bar{T}}. \tag{4.36}$$

By equating coefficients of increasing powers of z, matrix polynomial equation (4.31) is equivalent to the linear system

$$\langle A(z)\rangle_{d_U}\bar{U} = \bar{T}. \tag{4.37}$$

By construction, triangularization of $\boldsymbol{R}_d$ provides a triangular basis for submodule $\mathcal{R}_d$ from which triangular matrix $T(z)$ can be built. More precisely, suppose that $\boldsymbol{U}$ is a non-singular matrix such that

$$\boldsymbol{R}_d\boldsymbol{U} = \boldsymbol{T} \tag{4.38}$$

has triangular shape $\bar{\Sigma} = \{\bar{\sigma}_1, \ldots, \bar{\sigma}_{m(d_U+1)}\}$. Define column index sets

$$\mathcal{C}_i = \{k : i(d+1) - d \leq \bar{\sigma}_k \leq i(d+1)\}, \quad i = 1, \ldots, n. \tag{4.39}$$

With these notations, a matrix $\bar{U}$ such that $\mathrm{col}_i\bar{U} = \mathrm{col}_{k_i}\boldsymbol{U}$ for any $k_i \in \mathcal{C}_{\sigma_i}$, $i = 1, \ldots, m$ is solution to (4.37). Indeed each column $\mathrm{col}_{k_i}\boldsymbol{T}$ generates a column vector of shape $\{\sigma_i\}$ in $\mathcal{R}_d$, that is to say, a column $\mathrm{col}_i T(z)$ of a triangular form of $A(z)$ over $\mathbb{R}(z)$. $\square$

Lemma 4.4 states that if the degree d_U of a matrix $U(z)$ that reduces $A(z)$ into triangular form $T(z)$ is given, matrices $U(z)$ and $T(z)$ can be derived from triangularization (4.38). However, one usually does not know d_U beforehand, so let

$$\hat{d} = \sum_{i=1}^{m} \deg \mathrm{col}_i A(z) + \max_{i=1,\ldots,m} \deg \mathrm{col}_i A(z) - \min_{i=1,\ldots,m} \deg \mathrm{col}_i A(z). \tag{4.40}$$

Combination of Lemma 4.3 and Lemma 4.4 yields the following result.

Theorem 4.3 *Triangularizing over $\mathbb{R}(z)$ a full column rank polynomial matrix $A(z)$ is equivalent to triangularizing over $\mathbb{R}$ the row permuted Sylvester matrix $\boldsymbol{R}_{\hat{d}}$.*

Proof Note that $\hat{d} = d_A + \hat{d}_U$. Existence of a matrix $U(z)$ of degree d_U greater than or equal to $\hat{d}_U$ that brings $A(z)$ into triangular form is guaranteed by Lemma 4.3. Equivalence between triangularization of $A(z)$ over $\mathbb{R}(z)$ and triangularization of $\boldsymbol{R}_{\hat{d}}$ over $\mathbb{R}$ is stated in Lemma 4.4. $\square$

Corollary 4.3 *Suppose that a polynomial matrix $A^\dagger(z)$ of degree d_A does not necessarily have full column rank. Build the row permuted Sylvester matrix $\boldsymbol{R}_{\hat{d}}$ corresponding to the full column rank compound matrix $A(z)$ in (4.32). Triangularization of $\boldsymbol{R}_{\hat{d}}$ over $\mathbb{R}$ allows to evaluate the rank of $A^\dagger(z)$ and compute a polynomial basis for the polynomial right null-space of $A^\dagger(z)$.*

Proof Let d_A and m respectively denote the degree and the number of columns of $A(z)$. According to Theorem 4.2 and the proof of Lemma 4.4, triangularization (4.38) yields the triangular shape Σ of $T(z)$, a triangular form of $A(z)$ over $\mathbb{R}(z)$. The rank of $A^\dagger(z)$ is then readily the number of components of Σ less than or equal to the number of rows of $A^\dagger(z)$. Similarly, the $m - \operatorname{rank} A^\dagger(z)$ rightmost columns of $U(z)$ in (4.32) generate a polynomial right null-space of $A^\dagger(z)$ of degree less than or equal to $\hat{d}_U = \hat{d} - d_A$. $\square$

Remark 4.3 Corollary 4.3 is an alternative to the method proposed in [5, 6] where the null-space of a polynomial matrix is computed via a matrix pencil approach.

It must be noticed that Lemma 4.3 only provides an upper bound $\hat{d}_U$ on d_U. Hence, a reduction matrix $U(z)$ of degree d_U strictly less than $\hat{d}_U$ may exist. Since the value of d_U is generally unknown, one can alternatively

- triangularize $\boldsymbol{R}_{\hat{d}}$ only once, or
- triangularize $\boldsymbol{R}_d$ successively for $d = d_A + d_U$ and $d_U = 0, 1, \ldots, \hat{d}_U$ until a matrix $U(z)$ of minimal possible degree d_U is found.

Algorithm TriRatSyl **: Triangularization over $\mathbb{R}(z)$**
Sylvester matrix approach

Input A full column rank polynomial matrix $A(z) \in \mathbb{R}^{n\times m}[z]$ of degree d_A.
Output A non-singular reduction matrix $U(z) \in \mathbb{R}^{m\times m}[z]$ such that $A(z)U(z) = T(z)$ is a triangular form of $A(z)$ over $\mathbb{R}(z)$.

Step 0 Let $d_U = \hat{d}_U$ as in (4.33).
Step 1 Build permuted Sylvester matrix $\boldsymbol{R}_d$ as in (4.34), where $d = d_A + d_U$.
Step 2 Triangularize $\boldsymbol{R}_d$ over $\mathbb{R}$ as in (4.38).
Step 3 Extract a triangular shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ such that sets $\mathcal{C}_{\sigma_i}$ in (4.39) are not empty.
Step 4 Build matrix $\bar{U}$ such that $\mathrm{col}_i \bar{U} = \mathrm{col}_{k_i} \boldsymbol{U}$ for any $k_i \in \mathcal{C}_{\sigma_i}$, $i = 1, \ldots, m$.
Step 5 Recover $U(z)$ as in (4.36).

Algorithm TriRatSyl2 **:Triangularization over** $\mathbb{R}(z)$
Iterative Sylvester matrix approach

Only the following steps differ from Algorithm TriRatSyl

Step 0 Let $d_U = 0$.
Step 3 Extract a triangular shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ such that sets $\mathcal{C}_{\sigma_i}$ in (4.39) are not empty. If it is not possible, let $d_U = d_U + 1$ and go to Step 1.

Interpolation approach

An interpolation technique for triangularizing a polynomial matrix over $\mathbb{R}(z)$ is now derived. Relationships with the Sylvester matrix approach and the standard interpolation procedure described in [2] are examined.

In this section, it is supposed that matrix coefficients of $A(z) \in \mathbb{R}^{m\times n}[z]$ are not known. The polynomial matrix is rather given through constant values $A(z_i) \in \mathbb{R}^{m\times n}$ taken at distinct interpolation points $z_i \in \mathbb{R}$, $i = 0, 1, \ldots, d$. Furthermore, it is assumed that column degrees, or at least the degree d_A of $A(z)$ are known.

Matrix polynomial equation (4.31) can be written

$$L(z)\bar{U} = T(z) \tag{4.41}$$

where $\bar{U}$ is defined in (4.37) and

$$L(z) = [z^{d_U} \ \cdots \ z \ \ 1] \otimes A(z). \tag{4.42}$$

Using relation (4.42) and values $A(z_i)$, build the row permuted interpolated matrix

$$\boldsymbol{L}_d = P \begin{bmatrix} z_d^{d_U} A(z_d) & \cdots & z_d A(z_d) & A(z_d) \\ \vdots & & & \vdots \\ z_1^{d_U} A(z_1) & \cdots & z_1 A(z_1) & A(z_1) \\ z_0^{d_U} A(z_0) & \cdots & z_0 A(z_0) & A(z_0) \end{bmatrix} = P \begin{bmatrix} L(z_d) \\ \vdots \\ L(z_1) \\ L(z_0) \end{bmatrix}. \tag{4.43}$$

Lemma 4.5 *Triangularizing over $\mathbb{R}(z)$ a full column rank polynomial matrix $A(z)$ of degree d_A by a reduction matrix $U(z)$ of degree d_U amounts to triangularizing over $\mathbb{R}$ the row permuted interpolated matrix $\boldsymbol{L}_d$, where $d = d_A + d_U$.*

Proof One can easily check that matrix $\boldsymbol{L}_d$ is a Sylvester matrix of order d_U multiplied to the left by a block Vandermonde matrix and a permutation matrix P, viz.

$$\boldsymbol{L}_d = P(M_d \otimes \mathbf{I}_m)\langle A(z)\rangle_{d_U} \tag{4.44}$$

where

$$M_d = \begin{bmatrix} z_d^d & \cdots & z_d & 1 \\ \vdots & & & \vdots \\ z_1^d & \cdots & z_1 & 1 \\ z_0^d & \cdots & z_0 & 1 \end{bmatrix}.$$

Note that Vandermonde matrix M_d is invertible since the z_i are supposed to be distinct. Using relation 4.34), one gets

$$\boldsymbol{L}_d = (\mathbf{I}_m \otimes M_d)P\langle A(z)\rangle_{d_U} = (\mathbf{I}_m \otimes M_d)\boldsymbol{R}_d.$$

With the notations of the proof of Lemma 4.3, matrix $\boldsymbol{L}_d$ generate submodule $\mathcal{R}_d$ in the basis

$$\{\{z^d e_1, \ldots, ze_1, e_1\}M_d^{-1}, \ldots, \{z^d e_n, \ldots, ze_n, e_n\}M_d^{-1}\}.$$

Triangularization of $\boldsymbol{L}_d$ thus provides a triangular basis for submodule $\mathcal{R}_d$ from which a triangular matrix $T(z) = A(z)U(z)$ of shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ can be built. Suppose that $\boldsymbol{U}$ is a non-singular matrix such that

$$\boldsymbol{L}_d\boldsymbol{U} = \boldsymbol{T} \tag{4.45}$$

has triangular shape $\bar{\Sigma} = \{\bar{\sigma}_1, \ldots, \bar{\sigma}_{m(d_U+1)}\}$. Define column index sets $\mathcal{C}_{\sigma_i}$ as in (4.39). As in the proof of Lemma 4.3, a matrix $\bar{U}$ such that $\mathrm{col}_i\bar{U} = \mathrm{col}_{k_i}\boldsymbol{U}$ for any $k_i \in \mathcal{C}_{\sigma_i}$, $i = 1, \ldots, m$ is solution to (4.41). □

Remark 4.4 Antsaklis and Gao [2] proposed an interpolation technique to solve polynomial matrix equation (4.31) where $U(z)$ has to be found and $A(z), T(z)$ are known through interpolated values $A(z_i), T(z_i)$. The triangularization problem faced here is quite different since $U(z)$ is unknown, interpolated values $A(z_i)$ are given but interpolated values $T(z_i)$ are unknown. However, the way matrix $\boldsymbol{L}_d$ is built in (4.43) parallels the development leading to Theorem 3.1 in [2]. More specifically, equations (4.41), (4.45) in this chapter and equations (3.7), (3.9) in [2] are respectively equivalent. Following the notations of [2], each row of matrix $\boldsymbol{L}_d$ corresponds to a special choice of interpolation triplets (a_j, b_j, c_j) such that

$$\mathrm{row}_j\boldsymbol{L} = a_j L(c_j) = b_j, \quad j = 1, 2, \ldots, n(d+1)$$

where

$$\begin{array}{ll} a_1 = a_2 = \ldots = a_n = e_1 & c_1 = c_{n+1} = \ldots = c_{nd+1} = z_d \\ a_{n+1} = a_{n+2} = \ldots = a_{2n} = e_2 & c_2 = c_{n+2} = \ldots = c_{nd+2} = z_{d-1} \\ \vdots & \vdots \\ a_{nd+1} = a_{nd+2} = \ldots = a_{n(d+1)} = e_n & c_n = c_{2n} = \ldots = c_{n(d+1)} = z_0. \end{array}$$

Remark 4.5 Matrix polynomial equation (4.31) evaluated at interpolation points s_0, $s_1, \ldots, s_d$ reads

$$A(z_i)U(z_i) = T(z_i), \quad i = 0, 1, \ldots, d$$

where the $A(z_i)$ are given and $U(z)$, $T(z)$ are to be found. If one intents to compute $U(z)$ and $T(z)$ by

1. computing matrices $\boldsymbol{U}_i$ such that matrices $A(z_i)\boldsymbol{U}_i = \boldsymbol{T}_i$ have the same triangular shape Σ for all $i = 0, 1, \ldots, d$
2. recovering matrices $\hat{U}(z)$, $\hat{T}(z)$ from interpolation relations $\hat{U}(z_i) = \boldsymbol{U}_i$, $\hat{T}(z_i) = \boldsymbol{T}_i$, $i = 0, 1, \ldots, d$,

then $\hat{T}(z)$ has indeed triangular shape Σ but generally the matrix polynomial equation $A(z)\hat{U}(z) = \hat{T}(z)$ does not hold. In other words, triangularization of a polynomial matrix $A(z)$ over $\mathbb{R}(z)$ *is not equivalent to successive triangularizations* of interpolated matrices $A(z_i)$ over $\mathbb{R}$. According to matrix polynomial equation (4.41), $U(z)$ is actually derived from a constant matrix $\bar{U}$ solution to

$$L(z_i)\bar{U} = T(z_i), \quad i = 0, 1, \ldots, d.$$

Triangularization of $A(z)$ over $\mathbb{R}(z)$ is therefore *equivalent to a simultaneous triangularization* of interpolated matrices $L(z_i)$ over $\mathbb{R}$ by the same reduction matrix $\bar{U}$.

Lemma 4.5 states that once d_A and d_U are given, interpolated values $A(z_i)$, $i = 0, 1, \ldots, d = d_A + d_U$ can be used to derive a reduction matrix $U(z)$ and a triangular form $T(z)$ of $A(z)$. However, the degree of reduction matrix $U(z)$ is generally not known beforehand. As a result, one usually does not know how many interpolation points are required. The following theorem provides an upper bound.

Theorem 4.4 *Triangularizing over $\mathbb{R}(z)$ a full column rank polynomial matrix $A(z)$ amounts to triangularizing over $\mathbb{R}$ the row permuted interpolated matrix $\boldsymbol{L}_{\hat{d}}$.*

Proof Combine Lemma 4.3 and Lemma 4.5. □

In view of Lemma 4.5 and Theorem 4.4, the number of interpolated values $A(z_0), \ldots, A(z_d)$ that are needed to triangularize $A(z)$ lies within the interval $d_A \leq d \leq \hat{d}$. Since a reduction matrix may be found for a value of d strictly less than $\hat{d}$, one can alternatively

- triangularize $\boldsymbol{L}_{\hat{d}}$ only once, or
- triangularize $\boldsymbol{L}_d$ successively for $d = d_A, d_A + 1, \ldots, \hat{d}$ until $U(z)$ is found for the minimal possible value of d.

Algorithm TriRatInt **: Triangularization over** $\mathbb{R}(z)$
Interpolation Approach

Input Values $A(z_i) \in \mathbb{R}^{n\times m}$ of a polynomial matrix $A(z) \in \mathbb{R}^{n\times m}[z]$, interpolated at distinct points $z_i \in \mathbb{R},\ i = 0, 1, 2, \ldots$

Output A non-singular reduction matrix $U(z) \in \mathbb{R}^{m\times m}[z]$ such that $A(z)U(z) = T(z)$ is a triangular form of $A(z)$ over $\mathbb{R}(z)$.

Step 0 Let $d = \hat{d}$ as in (4.40).
Step 1 Build interpolated matrix $\boldsymbol{L}_d$ as in (4.43).
Step 2 Triangularize $\boldsymbol{L}_d$ over $\mathbb{R}$ as in (4.45)
Steps 3–4 See Algorithm TriRatSyl.

Algorithm TriRatInt2 **: Triangularization over** $\mathbb{R}(z)$
Iterative interpolation approach

Only the following steps differ from Algorithm TriRatInt.

Step 0 Let $d = d_A$.
Step 3 Extract triangular shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ such that sets $\mathcal{C}_{\sigma_i}$ in (4.39) are not empty. If it is not possible, let $d = d + 1$ and go to Step 1.

4.4.4 Numerical considerations

Algorithms TriRatSyl and TriRatInt only rely upon numerically reliable procedures. Indeed, in both algorithms the crucial point is Step 2, where a matrix has to be triangularized over field $\mathbb{R}$. This triangularization can be performed using powerful and reliable numerical procedures such as Householder, modified Gram-Schmidt or Givens orthogonalizations – the so-called QR decomposition, where the reduction matrix is orthogonal [15].

The conditioning of interpolated matrix $\boldsymbol{L}_d$ in Step 1 of Algorithm TriRatInt clearly depends on the choice of interpolation points and more specifically on the conditioning of Vandermonde matrix M_d – see equation (4.44). If interpolation points $z_0, z_1, \ldots, z_d$ lie on the real axis then the condition number of M_d grows at least exponentially with d. The slower growth rate is achieved if the z_i are the zeros of the $(d+1)$-th degree Chebyshev polynomial. If the z_i are the complex roots of unity, M_d is perfectly conditioned [16, §21.1] and Algorithm TriRatSyl and Algorithm TriRatInt are numerically equivalent. Note however that the possibility of using complex interpolation points stays out of the scope of this work and is subject to current research.

How to choose between non-iterative Algorithm TriRatSyl and iterative Algorithm TriRatSyl2 will depend on the problem being treated. Suppose that there exists a reduction matrix of low degree $d_U^\star \ll \hat{d}_U$. Since matrix $\boldsymbol{R}_d$ is $n(d_A + d_U + 1)$ by $m(d_U + 1)$, a single triangularization for a high value of $d_U = \hat{d}_U$ can easily be more demanding that several triangularizations for low values of $d_U = 0, 1, \ldots, d_U^\star$. The same comment holds for Algorithms TriRatInt and TriRatInt2, matrix $\boldsymbol{L}_d$ and parameter d.

In Algorithms TriRatSyl2 and TriRatInt2 matrices of increasing dimensions are triangularized until the lowest degree reduction matrix is found. At each step, the algorithms do not exploit the results computed at previous steps. As far as computational load is concerned, this may be considered as a drawback. At the other hand, it prevents the accumulation of rounding errors from step to step.

Finally, it must be noticed that for high values of n, m or d, matrices $\boldsymbol{R}_d$ and $\boldsymbol{L}_d$ may have large dimensions and unavoidable rounding errors may affect triangularizations (4.38) and (4.45). In that case, the notion of a shape of a polynomial matrix should preferably be replaced with the more realistic notion of an ε-shape, defined below.

Definition 4.6 *The ε-shape of a full column rank n-by-m matrix A is the series $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ of positive integers $\sigma_i = 1, 2, \ldots, n$ such that $\mathrm{ent}_{\sigma_i,i}A$ is the uppermost non-zero entry in $\mathrm{col}_i A$ of norm greater than ε.*

Step 3 of Algorithms TriRatSyl, TriRatSyl2, TriRatInt and TriRatInt2 must be modified accordingly. The value of ε should depend on n, m, d and the floating point relative accuracy of the computer.

4.4.5 Triangular Form over $\mathbb{R}[z]$

The algorithms described in the preceding section were designed to compute a reduction matrix $U(z)$ invertible over $\mathbb{R}(z)$, but not necessarily unimodular. In the sequel it is explained how to extend the Sylvester matrix and interpolation triangularization algorithms over $\mathbb{R}(z)$ so as to derive a unimodular reduction matrix. The instrumental key concept that will be introduced is polynomial degree minimization over ideals.

Sylvester matrix approach

Let $\mathcal{E}^j \subset \mathbb{R}^n[z]$ denote the submodule generated by $\{e_j, \ldots, e_n\}$. Let the map $p^j(.) : \mathbb{R}^n[z] \to \mathbb{R}[z]$ extract the j-th component of a polynomial module and define the submodule $\mathcal{A}^j = p^j(\mathcal{A} \cap \mathcal{E}^j)$. Note that $\mathcal{A}^j$ is an ideal over $\mathbb{R}[z]$. The following theorem algebraically characterizes triangular forms over $\mathbb{R}(z)$ and $\mathbb{R}[z]$, as defined in Theorem 4.2.

Theorem 4.5 *Given a full column rank polynomial matrix $A(z) \in \mathbb{R}^{n\times m}[z]$, let $T(z)$ stand for a matrix of triangular shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ such that $\mathrm{col}_i T(z) \in \mathcal{A} \cap \mathcal{E}^{\sigma_i}$. Then $T(z)$ is a triangular form of $A(z)$ over $\mathbb{R}(z)$. Moreover, if each leading entry $\mathrm{ent}_{\sigma_i,i}T(z)$*

has minimal degree in the ideal $\mathcal{A}^{\sigma_i}$ *for* $i = 1, \ldots, m$, *then* $T(z)$ *is a triangular form of* $A(z)$ *over* $\mathbb{R}[z]$.

Proof See [39, Proposition 1]. $\square$

The above theorem states that the triangular form of a polynomial matrix $A(z)$ over $\mathbb{R}[z]$ is actually a special triangular form of $A(z)$ over $\mathbb{R}(z)$ with the additional requirement that leading entry degrees must be minimal. Clearly, the triangular form of $A(z)$ over $\mathbb{R}(z)$ as computed by Algorithm TriRatSyl does not necessarily verify these constraints. The sequel aims at proving that Algorithm TriRatSyl can be extended for that purpose.

Suppose that a triangular form of $A(z)$ over $\mathbb{R}(z)$ of shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ has been found for a given value of d. Define the submodule $\mathcal{I}_d^{\sigma_i} = p^{\sigma_i}(\mathcal{R}_d \cap \mathcal{E}^{\sigma_i}) \subset \mathcal{A}^{\sigma_i}$. One can check that $\mathcal{I}_d^{\sigma_i} = \{0\}$ as soon as σ_i does not belong to Σ. According to Theorem 4.5, in order to derive a triangular form over $\mathbb{R}[z]$ it is necessary to find the minimal degree polynomials in ideals $\mathcal{I}_d^{\sigma_i}\mathbb{R}[z]$.

Lemma 4.6 *Given a full column rank polynomial matrix* $A(z)$ *of degree* d_A, *suppose that triangularization Algorithm* TriRatSyl *resulted in a triangular form of* $A(z)$ *over* $\mathbb{R}(z)$ *with a reduction matrix of degree* d_U, *and let* $d = d_A + d_U$. *Then, for any* $i = 1, \ldots, m$ *the minimal degree polynomial of ideal* $\mathcal{I}_d^{\sigma_i}\mathbb{R}[z]$ *belongs to submodule* $\mathcal{I}_d^{\sigma_i}$.

Before the exposition of the proof of Lemma 4.6, the following fundamental result [39, §3.3] is recalled.

Lemma 4.7 *Let* $\mathcal{H}_0$ *be a submodule in* $\mathbb{R}[z]_d$ *and let the iteration* $\mathcal{H}_{k+1} = z\mathcal{H}_k + \mathcal{H}_k$ *define a submodule sequence in* $\mathbb{R}[z]$ *for* $k \geq 0$. *For a given* k, *if* $\dim \mathcal{H}_{k+1} = \dim \mathcal{H}_{k+1}$ *then the minimum degree polynomial of ideal* $\mathcal{H}_0\mathbb{R}[z]$ *belongs to submodule* $\mathcal{H}_k$.

Proof of Lemma 4.7 : For any $k \geq 0$, $\mathcal{H}_{k+1}$ contains $\mathcal{H}_k$ and at least one polynomial of higher degree than any polynomial of $\mathcal{H}_k$, whence $\dim \mathcal{H}_{k+1} \geq \dim \mathcal{H}_k + 1$. If $\dim \mathcal{H}_{k+1} = \dim \mathcal{H}_k + 1$ holds for a given value of k, then the equality also holds for higher values of k, thus showing that a minimal degree polynomial was already found in $\mathcal{H}_k$. $\square$

Proof of Lemma 4.6 : By construction $\mathcal{R}_{d+1} = z\mathcal{R}_d + \mathcal{R}_d$ and therefore $\mathcal{I}_{d+1}^{\sigma_i} \subset z\mathcal{I}_d^{\sigma_i} + \mathcal{I}_d^{\sigma_i}$ for $i = 1, \ldots, m$ and any $d \geq d_A$. Triangularization of $\boldsymbol{R}_d$ resulted in a triangular shape $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, so $\dim \mathcal{R}_d = \sum_{i=1}^m \dim \mathcal{I}_d^{\sigma_i} = \operatorname{rank} \boldsymbol{R}_d$. Since $A(z)$ has full column rank m, $\operatorname{rank} \boldsymbol{R}_{d+1} = \operatorname{rank} \boldsymbol{R}_d + m$, hence $\dim \mathcal{R}_{d+1} = \sum_{i=1}^m \dim \mathcal{I}_{d+1}^{\sigma_i} = \sum_{i=1}^m \dim \mathcal{I}_d^{\sigma_i} + m$ and $\dim \mathcal{I}_{d+1}^{\sigma_i} = \dim \mathcal{I}_d^{\sigma_i} + 1$. According to Lemma 4.7, the minimal degree polynomial of ideal $\mathcal{I}_d^{\sigma_i}\mathbb{R}[z]$ can be found in submodule $\mathcal{I}_d^{\sigma_i}$. $\square$

Lemma 4.6 provides the theoretical guarantee that one can extract minimal degree polynomials by working directly on matrix $\boldsymbol{R}_d$. Let the sets $\mathcal{C}_i$ be as in (4.39). Define

$\boldsymbol{R}_d^{\sigma_i} = \{\mathrm{col}_j \boldsymbol{R}_d \cap \mathrm{row}_k \boldsymbol{R}_d$ for all $j \in \mathcal{C}_{\sigma_i}$ and all $k \in [\sigma_i(d+1)-d, \sigma_i(d+1)]\}$. Submodule $\mathcal{I}_d^{\sigma_i}$ is therefore generated by matrix $\boldsymbol{R}_d^{\sigma_i}$ in the basis $\{z^d, \ldots, z, 1\}$. Since $\boldsymbol{R}_d$ is triangular, so is $\boldsymbol{R}_d^{\sigma_i}$. As a result, the rightmost column of matrix $\boldsymbol{R}_d^{\sigma_i}$, i.e. the column indexed by the maximal element in $\mathcal{C}_i$, is a representation of the minimal degree polynomial in $\mathcal{I}_d^{\sigma_i}$. The following theorem has been shown.

Theorem 4.6 *Triangularizing over $\mathbb{R}[z]$ a full column rank polynomial matrix $A(z)$ amounts to triangularizing over $\mathbb{R}$ the row permuted Sylvester matrix $\boldsymbol{R}_{\hat{d}}$.*

Finally, the following corollary to Theorem 4.6 underlines the fact that the triangularization techniques presented here can also be used for greatest common divisor extraction.

Corollary 4.4 *Let $A(z) = [B_1(z) \ \cdots \ B_p(z)]$ be a full column rank polynomial matrix. Extraction of the greatest common left divisor of matrices $B_1(z), \ldots, B_p(z)$ is equivalent to triangularization over $\mathbb{R}$ of row permuted Sylvester matrix $\boldsymbol{R}_{\hat{d}}$.*

Proof See Theorem 4.2 and, by example, Lemma 6.3-3 in [22]. □

Algorithm TriPolSyl **: Triangularization over $\mathbb{R}[z]$**
Sylvester matrix approach

Input A full column rank polynomial matrix $A(z) \in \mathbb{R}^{n\times m}[z]$ of degree d_A.
Output A unimodular reduction matrix $U(z) \in \mathbb{R}^{m\times m}[z]$ such that $A(z)U(z) = T(z)$ is a triangular form of $A(z)$ over $\mathbb{R}[z]$.

Only the following step differs from Algorithm 1.

Step 4 Extract matrix $\bar{U}$, such that $\mathrm{col}_i \bar{U} = \mathrm{col}_{\hat{k}_i} \boldsymbol{U}$, where $\hat{k}_i$ is the maximal element in $\mathcal{C}_{\sigma_i}$, $i = 1, \ldots, m$.

Algorithm TriPolSyl2 **:Triangularization over $\mathbb{R}[z]$**
Iterative Sylvester matrix approach

Only steps 0 and 3, taken from Algorithm TriRatSyl2, differ from Algorithm TriPolSyl

Interpolation approach

Since interpolated matrix $\boldsymbol{L}_d$ and permuted Sylvester matrix $\boldsymbol{R}_d$ are equivalent up to a change of basis, the interpolation approach to triangularization over $\mathbb{R}[z]$ naturally follows.

Lemma 4.8 *Given interpolated values $A(z_0), A(z_1), \ldots, A(z_d)$, suppose that triangularization Algorithm* TRIRATINT *resulted in a triangular form of $A(z)$ over $\mathbb{R}(z)$. Then, for any $i = 1, \ldots, m$ the minimal degree polynomial of ideal $\mathcal{I}_d^{\sigma_i}\mathbb{R}[z]$ belongs to submodule $\mathcal{I}_d^{\sigma_i}$.*

Proof Similar to that of Lemma 4.6. □

Lemma 4.8 provides the guarantee that it is of no use to work with a higher number of interpolation points once a triangular form over $\mathbb{R}(z)$ is obtained. Therefore one can extract the minimal degree polynomials in $\mathcal{I}_d^{\sigma_i}$ by working directly on the matrix $\boldsymbol{T}$ derived by triangularization of $\boldsymbol{L}_d$. Let the sets $\mathcal{C}_{\sigma_i}$ be defined as in (4.39). Define $\boldsymbol{L}_d^{\sigma_i} = \{\mathrm{col}_j \boldsymbol{L}_d \cap \mathrm{row}_k \boldsymbol{L}_d$ for all $j \in \mathcal{C}_{\sigma_i}$ and all $k \in [\sigma_i(d+1) - d, \sigma_i(d+1)]\}$. Submodule $\mathcal{I}_d^{\sigma_i}$ is generated by matrix $\boldsymbol{L}_d^{\sigma_i}$ in the basis $\{z^d, \ldots, z, 1\}M_d^{-1}$. Thus an easy way to minimize the degree of a polynomial in $\mathcal{I}_d^{\sigma_i}$ could be to triangularize matrix $M_d^{-1}\boldsymbol{L}_d^{\sigma_i}$. The minimal degree polynomial then corresponds to the rightmost column of the computed triangular form. However, in doing so one does not take advantage of the special structure of interpolated matrix $\boldsymbol{L}_d^{\sigma_i}$. Moreover, Vandermonde matrix M_d may be ill-conditioned. Therefore, the use of a special Vandermonde linear system solver is recommended. Only a simplified version of the two-step algorithm exposed in [15] is required for degree minimization. It is described in the sequel.

For a given column index i, suppose that one aims at extracting the minimal degree polynomial generated by the columns of a matrix $\boldsymbol{I} = \boldsymbol{L}_d^{\sigma_i}$ in the basis $\{z^d, \ldots, z, 1\}M_d^{-1}$. Entries of $\boldsymbol{I}$ are denoted as follows

$$\boldsymbol{I} = \begin{bmatrix} t_1(z_d) & \cdots & t_p(z_d) \\ \vdots & & \vdots \\ t_1(z_1) & \cdots & t_p(z_1) \\ t_1(z_0) & \cdots & t_p(z_0) \end{bmatrix} \in \mathbb{R}^{(d+1)\times p}.$$

When $j = 1, \ldots, p$, each column of $\boldsymbol{I}$

$$\mathrm{col}_j \boldsymbol{I} = \begin{bmatrix} t_j(z_d) \\ \vdots \\ t_j(z_1) \\ t_j(z_0) \end{bmatrix} = M_d \begin{bmatrix} t_{d,j} \\ \vdots \\ t_{1,j} \\ t_{0,j} \end{bmatrix}$$

corresponds to a polynomial $t_j(z) = t_{0,j} + zt_{1,j} + \ldots + z^d t_{d,j}$ which Newton representation reads

$$t_j(z) = \sum_{k=0}^{d} c_{k,j} \prod_{l=0}^{k-1} (z - z_l).$$

Newton coefficients $c_{k,j}$ are divided differences and may be generated in an iterative way from entries of matrix $\boldsymbol{I}$ [15] to yield the Newton coefficient matrix

$$\boldsymbol{C} = \begin{bmatrix} c_{d,1} & \cdots & c_{d,p} \\ \vdots & & \vdots \\ c_{1,1} & \cdots & c_{1,p} \\ c_{0,1} & \cdots & c_{0,p} \end{bmatrix}.$$

Triangularization of $\boldsymbol{C}$ reads

$$\boldsymbol{F} = \boldsymbol{C}\boldsymbol{V} = \left[\begin{array}{c|c|c|c} f_{d,1} & 0 & \cdots & 0 \\ f_{d-1,1} & f_{d-1,2} & & \vdots \\ \vdots & \vdots & \ddots & 0 \\ f_{q,1} & f_{q,2} & & f_{q,p} \\ \vdots & \vdots & & \vdots \\ f_{1,1} & f_{1,2} & & f_{1,p} \\ f_{0,1} & f_{0,2} & \cdots & f_{0,p} \end{array}\right]. \tag{4.46}$$

The last column of $\boldsymbol{F}$ thus corresponds to the polynomial

$$t_{\min}(z) = \sum_{k=0}^{q} f_{k,p} \prod_{l=0}^{k-1} (z - z_l)$$

featuring the greatest number of zero leading coefficients, that is to say, to the expected minimal degree polynomial. The procedure is as follows.

Procedure DEGMIN : Degree minimization over interpolated polynomial submodule

Input Interpolation points $z_0, z_1, \ldots, z_d$ and a full column rank matrix $\boldsymbol{I}$ whose p columns generate a submodule of $\mathbb{R}[z]_d$ in the basis $\{z^d, \ldots, z, 1\}M_d^{-1}$.

Output The vector v such that $\boldsymbol{I}v$ is a representation w.r.t. the same basis of the minimal degree polynomial in this submodule.

Step 1 Build Newton matrix $\boldsymbol{C}$ using the iteration

$$\begin{array}{l} \boldsymbol{C} = \boldsymbol{I} \\ \text{for } k = 1, \ldots, d \\ \quad \text{for } i = 1, \ldots, d+1-k \\ \quad\quad \text{row}_i\boldsymbol{C} = (\text{row}_i\boldsymbol{C} - \text{row}_{i+1}\boldsymbol{C})/(z_{d+1-i} - z_{d+1-i-k}). \end{array}$$

Step 2 Triangularize matrix $\boldsymbol{C}$ over $\mathbb{R}$ as in (4.46). Vector v is the rightmost column in $\boldsymbol{V}$.

If Procedure DEGMIN is successively applied to matrices $\boldsymbol{L}_d^{\sigma_i}$ corresponding to submodules $\mathcal{I}_d^{\sigma_i}$ for $i = 1, \ldots, m$, one gets a reduction matrix $U(z)$ such that degrees of polynomials $\text{ent}_{\sigma_i,i}A(z)U(z)$ are minimized. According to Theorem 4.5, $U(z)$ triangularizes $A(z)$ over $\mathbb{R}[z]$. The following theorem has been shown.

Theorem 4.7 *Triangularizing over $\mathbb{R}[z]$ a full column rank polynomial matrix $A(z)$ amounts to triangularizing over $\mathbb{R}$ the interpolated matrix $\boldsymbol{L}_{\hat{d}}$, followed by m degree minimizations over interpolated polynomial submodules as described in Procedure* DEGMIN, *where m is the number of columns of $A(z)$.*

Algorithm TRIPOLINT **: Triangularization over $\mathbb{R}[z]$**
Interpolation approach

Input Values $A(z_i) \in \mathbb{R}^{n\times m}$ of a polynomial matrix $A(z) \in \mathbb{R}^{n\times m}[z]$, interpolated at distinct points $z_i \in \mathbb{R},\ i = 0, 1, 2, \ldots$.

Output A unimodular reduction matrix $U(z) \in \mathbb{R}^{m\times m}[z]$ such that $A(z)U(z) = T(z)$ is a triangular form of $A(z)$ over $\mathbb{R}[z]$.

Only Step 4 differs from Algorithm TRIRATINT

Step 4.a Extract sub-matrices $\boldsymbol{L}_d^{\sigma_i} = \{\mathrm{col}_j \boldsymbol{L}_d \cap \mathrm{row}_k \boldsymbol{L}_d,$ for all $j \in \mathcal{C}_{\sigma_i}$ and all $k \in [\sigma_i(d+1) - d, \sigma_i(d+1)]\}$. Procedure DEGMIN with inputs $z_0, z_1, \ldots, z_d$ and $\boldsymbol{L}_d^{\sigma_i}$ generates outputs v_i for $i = 1, \ldots, m$.

Step 4.b Extract sub-matrices $\boldsymbol{U}^{\sigma_i} = \{\mathrm{col}_j \boldsymbol{U}^{\sigma_i}$ for all $j \in \mathcal{C}_{\sigma_i}\}$. Build matrix $\bar{U}$ such that $\mathrm{col}_i \bar{U} = \boldsymbol{U}^{\sigma_i} v_i$ for $i = 1, \ldots, m$.

Algorithm TRIPOLINT2 **:Triangularization over $\mathbb{R}[z]$**
Iterative Interpolation approach

See Algorithm TRIPOLINT, except for Steps 0 and 3, see Algorithm TRIRATINT2.

4.4.6 Numerical considerations

Obviously, the derivation of Algorithm TRIPOLSYL from Algorithm TRIRATSYL does not introduce any potential numerical instability. Things are quite different in Algorithm TRIPOLINT since the column selection scheme in Step 4 hinges on Procedure DEGMIN. Hopefully, the computation of Newton coefficients in Step 1 of Procedure DEGMIN generally produces surprisingly accurate solutions even when Vandermonde matrix M_d is ill-conditioned – see [15] or [16, §21.3]. Furthermore, the triangularization in Step 2 of Procedure DEGMIN can be performed via powerful numerical methods such as column Householder transformations [15].

The column selection schemes in Step 4 of Algorithms TRIPOLSYL and TRIPOLINT ensure that the reduction matrix $U(z)$ is unimodular. In [8], the authors underlined the fact that their Gaussian elimination algorithm, in order to exploit the special structure of the Sylvester matrix, was designed to preserve unimodularity to the detriment of numerical stability. By contrast, the procedures described here hinge on orthogonal matrix operations while preserving unimodularity of the reduction matrix. One can naturally argue that the procedures do not take advantage of the special structure of the Sylvester or interpolated matrices, with the notable exception of Procedure DEGMIN. This seems to be the price one has to pay for numerical reliability, but does not appear to be a fundamental limitation in view of the availability of powerful computer tools.

Finally, if one is interested in getting the Hermite form defined in Theorem 4.2 rather than an arbitrary triangular form over $\mathbb{R}[z]$, this can straightforwardly be done using elementary column operations. Each leading entry must be made monic and entries at the left of it must be reduced via the Euclidean algorithm. Naturally, this may be achieved at the expense of numerical reliability. However, the number of potentially unstable operations needed for computing the Hermite form of a triangular form over $\mathbb{R}[z]$ is certainly far less than the number of operations needed for computing the Hermite form of a polynomial matrix in arbitrary form.

4.4.7 Numerical example

Both Sylvester matrix and interpolation approaches are now illustrated on a very simple example: reduction to lower left triangular form of polynomial matrix

$$A(z) = \begin{bmatrix} z-1 & z^2-1 \\ 2 & 2z+2 \\ 0 & 3 \end{bmatrix}.$$

Sylvester matrix approach

First, Algorithm TRIRATSYL2 will be used to derive a minimal degree reduction matrix. Let $d_U = 0$, $d = d_A + d_U = 2$ and built $\boldsymbol{R}_2$ as in (4.34). Triangularization (4.38) reads

$$\underbrace{\left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \\ -1 & -1 \\ \hline 0 & 0 \\ 0 & 2 \\ 2 & 2 \\ \hline 0 & 0 \\ 0 & 0 \\ 0 & 3 \end{array}\right]}_{\boldsymbol{R}_2} \underbrace{\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}}_{\boldsymbol{U}} = \underbrace{\left[\begin{array}{cc} -1 & 0 \\ 0 & 1 \\ 1 & -1 \\ \hline 0 & 0 \\ -2 & 0 \\ -2 & 2 \\ \hline 0 & 0 \\ 0 & 0 \\ -3 & 0 \end{array}\right]}_{\boldsymbol{T}}$$

Triangular matrix $\boldsymbol{T}$ has shape $\bar{\Sigma} = \{\bar{\sigma}_1, \bar{\sigma}_2\} = \{1, 2\}$. The sets $\mathcal{C}_i$ defined in (4.39) are $\mathcal{C}_1 = \{1, 2\}, \mathcal{C}_2 = \varnothing$ and $\mathcal{C}_3 = \varnothing$. Since $\mathcal{C}_2$ and $\mathcal{C}_3$ are empty, no reduction matrix of degree

$d_U = 0$ has been found. Letting $d_U = d_U + 1$, next triangularization (4.38) reads

$$\underbrace{\left[\begin{array}{cc|cc} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 \\ 0 & 0 & -1 & -1 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 2 & 2 & 0 & 2 \\ 0 & 0 & 2 & 2 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 \end{array}\right]}_{\boldsymbol{R}_3} \underbrace{\left[\begin{array}{ccc|c} 0 & -0.707 & 0.408 & -0.577 \\ -1 & 0 & 0 & 0 \\ \hline 0 & 0 & -0.817 & -0.577 \\ 0 & -0.707 & -0.408 & 0.577 \end{array}\right]}_{\boldsymbol{U}} = \underbrace{\left[\begin{array}{ccc|c} -1 & 0 & 0 & 0 \\ 0 & -1.414 & 0 & 0 \\ 1 & 0.707 & -1.225 & 0 \\ 0 & 0.707 & 1.225 & 0 \\ \hline 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ -2 & -2.828 & 0 & 0 \\ 0 & -1.414 & -2.450 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ 0 & -2.121 & -1.225 & 1.732 \end{array}\right]}_{\boldsymbol{T}}.$$

Now, the shape of triangular matrix $\boldsymbol{T}$ is $\bar{\Sigma} = \{1, 2, 3, 12\}$. Column index sets are $\mathcal{C}_1 = \{1, 2, 3\}, \mathcal{C}_2 = \varnothing$ and $\mathcal{C}_3 = \{12\}$ whence triangular shape $\Sigma = \{\sigma_1, \sigma_2\} = \{1, 3\}$. A possible choice for matrix $\bar{U}$ is for example $\bar{U} = [\mathrm{col}_1\boldsymbol{U} \;\; \mathrm{col}_4\boldsymbol{U}]$. The corresponding polynomial matrix is

$$U(z) = \begin{bmatrix} 0 & -0.577z - 0.577 \\ -z & 0.577 \end{bmatrix}.$$

One can check that $U(z)$ is not unimodular, so that

$$T(z) = A(z)U(z) = \begin{bmatrix} -z^3 + z & 0 \\ -2z^2 - 2z & 0 \\ -3z & 1.732 \end{bmatrix}$$

is only a triangular form of $A(z)$ over $\mathbb{R}(z)$. Now suppose that one is seeking a triangular form over $\mathbb{R}[z]$. From Algorithm TriPolSyl2, $\hat{k}_1 = \max\{k \in \mathcal{C}_1\} = 3$ and $\hat{k}_2 = \max\{k \in \mathcal{C}_3\} = 12$ so that matrix $\bar{U} = [\mathrm{col}_3\boldsymbol{U} \;\; \mathrm{col}_4\boldsymbol{U}]$ yields a unimodular matrix

$$U(z) = \begin{bmatrix} -0.817 + 0.408z & -0.577z - 0.577 \\ -0.408 & 0.577 \end{bmatrix}$$

and a triangular form

$$T(z) = A(z)U(z) = \begin{bmatrix} -1.225z + 1.225 & 0 \\ -2.450 & 0 \\ -1.225 & 1.732 \end{bmatrix}$$

of $A(z)$ over $\mathbb{R}[z]$.

Interpolation approach

Now assume that polynomial matrix $A(z)$ is only given through interpolated values $A(z_0), \ldots, A(z_d)$ and that Algorithm TriPolInt is used to compute a reduction matrix over $\mathbb{R}[z]$. Therefore, let $d = \hat{d} = 4$ as in (4.40). By example, choose interpolation points $z_0 = -2, z_1 = -1, z_2 = 0, z_3 = 1, z_4 = 2$. Algorithm TRIPOLINT directly leads to triangularization

$$
\left[\begin{array}{rr|rr|rr}
-12 & 12 & 6 & -6 & -3 & 3 \\
-2 & 0 & 2 & 0 & -2 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
4 & 12 & 2 & 6 & 1 & 3 \\
\hline
8 & -8 & -4 & 4 & 2 & -2 \\
2 & 0 & -2 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 2 & 2 \\
2 & 4 & 2 & 4 & 2 & 4 \\
8 & 24 & 4 & 12 & 2 & 6 \\
\hline
0 & 12 & 0 & -6 & 0 & 3 \\
0 & 3 & 0 & -3 & 0 & 3 \\
0 & 0 & 0 & 0 & 0 & 3 \\
0 & 3 & 0 & 3 & 0 & 3 \\
0 & 12 & 0 & 6 & 0 & 3
\end{array}\right]
\left[\begin{array}{rrrr|rr}
-0.617 & -0.246 & 0.151 & -0.401 & 0.568 & -0.229 \\
0.617 & -0.492 & 0.302 & -0.535 & 0 & 0 \\
\hline
0.309 & 0.492 & -0.302 & -0.267 & 0.162 & -0.688 \\
-0.309 & 0.246 & -0.151 & -0.668 & -0.568 & 0.229 \\
\hline
-0.154 & -0.616 & -0.452 & 0.134 & -0.406 & -0.459 \\
0.154 & -0.123 & -0.754 & -0.134 & 0.406 & 0.459
\end{array}\right] =
$$

$$
\boldsymbol{L}_4 \underbrace{[\boldsymbol{U}^{\sigma_1} \mid \boldsymbol{U}^{\sigma_2}]}_{\boldsymbol{U}} =
$$

$$
\left[\begin{array}{rrrr|rr}
19.44 & 0 & 0 & 0 & 0 & 0 \\
2.160 & 2.708 & 0 & 0 & 0 & 0 \\
0 & 0.739 & 1.206 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
4.012 & -5.416 & 0 & -12.83 & 0 & 0 \\
\hline
-12.96 & 0 & 0 & 0 & 0 & 0 \\
-2.160 & -2.708 & 0 & 0 & 0 & 0 \\
0 & -1.477 & -2.412 & 0 & 0 & 0 \\
0.926 & -2.216 & -3.618 & -6.414 & 0 & 0 \\
8.024 & -10.83 & 0 & -25.66 & 0 & 0 \\
\hline
9.721 & -7.755 & 2.261 & -2.806 & 4.623 & 0 \\
3.240 & -2.585 & -0.905 & 0 & 2.920 & 0.688 \\
0.463 & -0.369 & -2.261 & -0.401 & 1.217 & 1.377 \\
1.389 & -1.108 & -1.809 & -4.009 & -0.487 & 2.065 \\
6.018 & -4.801 & 0.452 & -10.82 & -2.190 & 2.753
\end{array}\right] = \boldsymbol{T}.
$$

Matrix $\boldsymbol{T}$ has shape $\bar{\Sigma} = \{1, 2, 3, 5, 11, 12\}$ hence $\mathcal{C}_1 = \{1, 2, 3, 5\}$, $\mathcal{C}_2 = \varnothing$, $\mathcal{C}_3 = \{11, 12\}$ and $\Sigma = \{\sigma_1, \sigma_2\} = \{1, 3\}$. In order to get a triangular form of $A(z)$ over $\mathbb{R}[z]$, two degree minimizations must be performed on matrices

$$
\boldsymbol{L}^{\sigma_1} = \left[\begin{array}{rrrr}
19.44 & 0 & 0 & 0 \\
2.160 & 2.708 & 0 & 0 \\
0 & 0.739 & 1.206 & 0 \\
0 & 0 & 0 & 0 \\
4.012 & -5.416 & 0 & -12.83
\end{array}\right] \text{ and } \boldsymbol{L}^{\sigma_2} = \left[\begin{array}{rr}
4.623 & 0 \\
2.920 & 0.688 \\
1.217 & 1.377 \\
-0.487 & 2.065 \\
-2.190 & 2.753
\end{array}\right].
$$

Using Procedure DegMin, one gets $v_1 = [-0.195 \ \ -0.779 \ \ -0.572 \ \ 0.169]^T$ and $v_2 = [0.375 \ \ 0.927]^T$. Finally, Algorithm TriPolSyl yields

$$\bar{U} = [\boldsymbol{U}^{\sigma_1} v_1 \ \ \boldsymbol{U}^{\sigma_2} v_2] = \left[\begin{array}{rr} 0.158 & 0 \\ 0 & 0 \\ \hline -0.316 & -0.577 \\ -0.158 & 0 \\ \hline 0.791 & -0.577 \\ 0.474 & 0.577 \end{array}\right]$$

and the unimodular reduction matrix

$$U(z) = \begin{bmatrix} 0.158z^2 - 0.316z + 0.791 & -0.577z - 0.577 \\ -0.158z + 0.474 & 0.577 \end{bmatrix}.$$

The corresponding triangular form of $A(z)$ over $\mathbb{R}[z]$ is

$$T(z) = A(z)U(z) = \begin{bmatrix} 1.265z - 1.265 & 0 \\ 2.530 & 0 \\ -0.474z + 1.423 & 1.732 \end{bmatrix}.$$

Note that the reduction matrix obtained here has degree two, whereas a unimodular reduction matrix of degree one has been found using iterative Algorithm TriPolSyl2.

4.4.8 Comparison with other triangularization methods

Numerical tests have been carried out to compare the performances of our algorithms with other triangularization methods. Only triangularization over $\mathbb{R}[z]$ has been considered. In addition to the algorithms presented here, namely

Algorithm A – Sylvester matrix approach (Algorithm TriPolSyl)

Algorithm B – Interpolation approach (Algorithm TriPolInt)

the other triangularization algorithms that were tested are

Algorithm C – Labhalla's Gaussian elimination algorithm [39, Algorithm Hermipol]

Algorithm D – Standard elementary polynomial operations (see Theorem 4.2)

Algorithm E – Maple[1] built-in symbolic routine `linalg[hermite]`.

Some remarks concerning implementation are listed below.

[1] Maple is a registered trademark of Waterloo Maple Software.

- The routines have been tested on square polynomial matrices of dimension n, degree d and random integer coefficients with absolute value strictly less than 10.

- The minimal possible degree of a matrix $U(z)$ that triangularizes a matrix $A(z)$ as above is generically equal to the upper bound $\hat{d}_U$ given in (4.33), namely $\hat{d}_U = d(n-1)$. For that reason, tests were not conducted on iterative Algorithms TriPolSyl2 and TriPolInt2. Their execution times would necessarily be greater than those of Algorithms TriPolSyl and TriPolInt, respectively. Note however that this may not be the case for triangularization problems arising from practical control issues. See also the example in the previous section.

- At the exception of Algorithm E, the algorithms have been implemented in Matlab 5.1 with floating point accuracy $2.22 \cdot 10^{-16}$ on a Sun SparcStation 4.

- Algorithms A and B rely upon a triangularization over $\mathbb{R}$ that is performed by column Householder transformations. They are entirely written in Matlab code and no external routine call is made. In particular, we did not use the Matlab built-in routine `qr`.

- Algorithm C relies upon a triangularization over $\mathbb{R}$ that is performed by successive applications of a modified Gaussian elimination with only partial pivoting, as described in [39]. It is also entirely written in interpreted Matlab code.

- Interpolation points in Algorithm B are equispaced within the real interval $[-1, +1]$.

- The value $\varepsilon = n \cdot d \cdot 10^{-10}$ is used to extract the ε-shape (see Definition 4.6) in Algorithms A and B.

- The criterion that has been used to decide whether the triangularization has properly been achieved is twofold.

  1. All coefficients in the strictly upper triangular part of the triangular form $T(z) = A(z)U(z)$ must have absolute value less than 10^{-8}.
  2. The inequalities
  $$0.9 < \frac{|\det U(z)|}{|\det U(0)|} < 1.1 \tag{4.47}$$
  must hold for any $z \in [-1, +1]$. This is aimed at testing unimodularity of $U(z)$ without computing $\det U(z)$, a numerically difficult task for high values of n and d.

Average execution times in seconds are reported in Table 1. A star ($\star$) means that the triangularization failed because the above criteria were not verified. A double star ($\star\star$) means that the execution time exceeded 1500 seconds.

		Algorithm				
d	n	A	B	C	D	E
3	3	0.21	1.3	0.62	16	0.65
5	3	0.51	2.4	1.1	26	2.7
7	3	0.87	4.0	1.6	39	12
9	3	1.5	6.1	2.3	54	30
3	4	0.46	2.4	1.1	29	8.8
5	4	2.0	5.6	2.5	59	111
7	4	4.1	10	3.7	$\star$	603
9	4	7.8	$\star$	5.5	$\star$	$\star\star$
3	5	2.1	4.6	2.3	$\star$	109
5	5	7.3	12	4.6	$\star$	$\star\star$
7	5	14.4	$\star$	8.0	$\star$	$\star\star$
9	5	29	$\star$	12	$\star$	$\star\star$
3	9	48	59	17	$\star$	$\star\star$
5	9	220	$\star$	$\star$	$\star$	$\star\star$
7	9	580	$\star$	$\star$	$\star$	$\star\star$
9	9	1200	$\star$	$\star$	$\star$	$\star\star$

Table 1 : Comparative execution times in seconds.

Some comments are in order.

- As expected, Algorithm D is the less reliable triangularization method. Even for relatively low values of d and n, Algorithm D returns a result that is generally extremely far from the correct result.

- Algorithm E is based on symbolic computation. This approach is known to be subject to intermediate expression swell and turns out to be impractical even for medium-sized problems [47, Chapter 3].

- Algorithm B behaves badly for relatively high values of d and n. This is due to an ill-conditioning of the interpolated matrix, in turn due to the choice of interpolation points.

- For low values of n, Algorithm A is faster than Algorithm C. The tendency is reversed when n is higher. In the latter case, Algorithm A seems to be more reliable than Algorithm C, to the expense of computational load.

4.5 Continuous-time symmetric equation

4.5.1 Introduction

Symmetric matrix polynomial equation of the form

$$A^T(-s)X(s) + X^T(-s)A(s) = 2B(s) \tag{4.48}$$

where $A(s)$ is a given square polynomial matrix, $B(s)$ is a para-Hermitian matrix verifying the symmetry property

$$B(s) = B^T(-s) \tag{4.49}$$

and $X(s)$ is a polynomial matrix to be found, arises in various areas of system control theory.

So polynomial matrix spectral factorization, a standard technique applied in many H_2 control and filtering problems – such as LQ optimal regulation, Kalman filtering, minimum variance LQG control – is traditionally handled by Newton-Raphson iterative scheme consisting of solution to equation (4.48) in every step. Its overall performance naturally hinges on the way equation (4.48) is processed. Next to its control applications, equation (4.48) is also encountered in the field of signal processing. There it is namely used in the polynomial approach to computation of an ARMA covariance function [45].

In a famous paper published in Automatica [19], a spectral factorization technique was presented that became standard: after the calculation of a polynomial adjoint matrix, the resolution of (4.48) can be reduced to the application of an Euclidean division algorithm (or Schur reduction) to a set of scalar polynomial equations. The availability of efficient numerical methods to compute the adjoint of a polynomial matrix [46] nevertheless contrasts with the poor stability properties of the reduction process.

Another algorithm based on triangularization was presented in [20] but only for the discrete-time case and also with relatively bad numerical properties since still limited to elementary polynomial operations.

In a more general context the interpolation approach, comprehensively exposed in [2], appears as a numerically reliable, flexible and easy-to-implement technique to solve unilateral matrix polynomial equations (including for instance the Diophantine equation). Their generalization of the scalar polynomial interpolation to the matrix case is shown to be particularly well suited to studying and solving a wide range of control problems. In a unified way, they provide new interpretations and demonstrations of well-known control theorems.

Finally, since equations involving just polynomials can also be solved by equating coefficients of equal powers of the indeterminate s, alternate methods of this type are extensively described in [46]. They are based on the Sylvester resultant formulation of the product of two polynomial matrices [11, 22]. The need of inversion of relatively high order constant matrices is not a drawback anymore, in view of the availability of powerful linear equation solvers.

In this section we aim to prove that the interpolation approach developed in [2] can be extended to handle the bilateral symmetric equation (4.48). It is first shown that, by taking advantage of symmetry property (4.49), para-Hermitian matrix $B(s)$ can be represented by a restricted number of interpolation points. Using this fact, an interpolation formulation of equation (4.48) is proposed that results in a characterization of all solutions to (4.48) of expected column degrees. Symmetry properties are handled to significantly reduce the original problem dimension. Some matrix manipulations are then needed to obtain a reduced Sylvester resultant matrix formulation of the bilateral symmetric equation. In addition to provide an alternative method to solve (4.48), it throws light on relations with polynomial stability theory and with Hurwitz matrices [4, 40]. Moreover, the use of interpolation theory mixed with rank properties of the reduced resultant matrix will lead under some assumptions on $A(s)$ to conditions for existence of solutions to (4.48). An algorithm based on the reduced resultant formulation is first derived to solve (4.48). It is then followed by two simple interpolation algorithms. Some basic examples underline the simplicity of the methods. Finally, numerical experiments are reported that aim at demonstrating that the proposed algorithms are more efficient and reliable than already existing methods.

As far as the authors are aware, this appears to be the first numerically reliable solution technique for (4.48) ever published. As another advantage, no a priori assumptions on $A(s)$ and $B(s)$ are necessary for the algorithm. Hence critical cases such as for instance the presence of purely imaginary values as roots of $A(s)$ are handled without taking special precautions. If, all the same, matrix $A(s)$ is known to be stable (this is always the case when applied for spectral factorization), the method gets even simpler as now a solution is guaranteed to exist.

4.5.2 Description using interpolation of para-Hermitian polynomial matrices

In this section two methods to represent a para-Hermitian polynomial matrix using interpolation points are discussed. They parallel Theorem 2.1 in [2] and take fully advantage of symmetry properties to reduce the original data dimensions.

For any n-by-n polynomial matrix $X(s)$, vec$X(s)$ is the polynomial column vector obtained by stacking together columns of $X(s)$.

Given an n-by-n para-Hermitian polynomial matrix $B(s) = B_0 + B_1 s + \cdots + B_{d_B} s^{d_B}$, the following operators are defined :

- $\text{vec}_{\text{t}} B$ is a column vector obtained by stacking for $k = 0, 1, \ldots, d_B$
  - the $n - i + 1$ last elements of the i-th column of B_k for $i = 1, 2, \ldots, n$ when k

is even (that is, the lower left triangular part of B_k)

– the $n-i$ last elements of the i-th column of B_{k+1} for $i = 1, 2, \ldots, n-1$ when k is odd (that is, the strictly lower left triangular part of B_{k+1})

- $\widehat{\mathrm{vec}}_{\mathrm{t}}\, B$ is a reduced column vector, derived from $\mathrm{vec}_{\mathrm{t}}\, B$ by suppressing all zero leading coefficients of polynomial entries of $B(s)$.

Example 4.2 *For instance, if*

$$B(s) = \begin{bmatrix} 1 & -4s+2 \\ 4s+2 & 5s^2+3 \end{bmatrix}$$

then $\mathrm{vec}_{\mathrm{t}}\, B = [1\ 2\ 3\ 4\ 0\ 0\ 5]^T$ *and* $\widehat{\mathrm{vec}}_{\mathrm{t}}\, B = [1\ 2\ 3\ 4\ 5]^T$.

Polynomial equation (4.49) can be arranged by ascending powers of s to give

$$(B_0 - B_0^T) + (B_1 + B_1^T)s + (B_2 - B_2^T)s^2 + \cdots = 0.$$

Hence for k even

$$\begin{array}{l} \bullet\ B_k \text{ is symmetric} \\ \bullet\ B_{k+1} \text{ is skew-symmetric.} \end{array} \tag{4.50}$$

Entries of $\mathrm{vec}B(s)$ are also entries of $B(s)$, so each component of $\mathrm{vec}B(s)$ can be written as the product of a polynomial vector basis by a constant vector of coefficients of $B(s)$. Since $\mathrm{vec}_{\mathrm{t}}\, B$ and $\widehat{\mathrm{vec}}_{\mathrm{t}}\, B$ consist of these coefficients, define $S(s)$ and $\widehat{S}(s)$ as the unique polynomial matrices such that for any $B(s)$,

$$\mathrm{vec}B(s) = S(s)\, \mathrm{vec}_{\mathrm{t}}\, B \tag{4.51}$$

and

$$\mathrm{vec}B(s) = \widehat{S}(s)\, \widehat{\mathrm{vec}}_{\mathrm{t}}\, B. \tag{4.52}$$

The above relations play the same role than equation (2.4) in [2].

Example 4.3 *For the matrix $B(s)$ given in Example 4.2, equation (4.51) reads*

$$\mathrm{vec}B(s) = \left[\begin{array}{ccc|c|ccc} 1 & & & & s^2 & & \\ & 1 & & s & & s^2 & \\ & 1 & & -s & & s^2 & \\ & & 1 & & & & s^2 \end{array}\right] \left[\begin{array}{c} 1 \\ 2 \\ 3 \\ \hline 4 \\ \hline 0 \\ 0 \\ 5 \end{array}\right] = S(s)\, \mathrm{vec}_{\mathrm{t}}\, B$$

and equation (4.52) becomes

$$\mathrm{vec}B(s) = \left[\begin{array}{ccc|c|c} 1 & & & & \\ & 1 & & s & \\ & 1 & & -s & \\ & & 1 & & s^2 \end{array}\right] \left[\begin{array}{c} 1 \\ 2 \\ 3 \\ \hline 4 \\ \hline 5 \end{array}\right] = \widehat{S}(s)\, \widehat{\mathrm{vec}}_{\mathrm{t}}\, B.$$

Finally for a fixed j let $a_j \neq 0$ denote an n^2-complex row vector and b_j, s_j complex scalars. With these notations, the following theorem provides an interpolation description of $B(s)$.

Theorem 4.8 *Given an n-by-n symmetric matrix $\Delta = [d_{ijB}]$ of non-negative integers with even diagonal elements and*

$$\hat{q} = \frac{1}{2}n(n+1) + \frac{1}{2}\sum_{i,j=1}^{n} d_{ijB}$$

interpolation triplets (s_k, a_k, b_k), $k = 1, 2, \ldots, \hat{q}$ such that the $\hat{q}$-by-$\hat{q}$ matrix

$$\widehat{S}_{\hat{q}} = \begin{bmatrix} a_1\widehat{S}(s_1) \\ a_2\widehat{S}(s_2) \\ \vdots \\ a_{\hat{q}}\widehat{S}(s_{\hat{q}}) \end{bmatrix} \tag{4.53}$$

is non-singular, there exists a unique n-by-n para-Hermitian matrix $B(s)$ with entry degrees d_{ijB} for which

$$a_k \mathrm{vec} B(s_k) = b_k \tag{4.54}$$

for $k = 1, 2, \ldots, \hat{q}$.

Proof This follows by direct application of basic interpolation Theorem 2.1 in [2]. Note that $\widehat{S}(s)$ only depends on Δ (not on $B(s)$) and is made of $n(n+1)/2 + \sum_{i,j=1}^{n} d_{ijB}/2$ columns and n^2 rows, hence justifying the number of interpolation points. Since Δ contains entry degrees of $B(s)$, substitution of (4.54) into (4.52) yields

$$\widehat{S}_{\hat{q}}\ \widehat{\mathrm{vec}}_{\mathrm{t}} B = [b_1\, b_2\ \cdots\ b_{\hat{q}}]^T = T_{\hat{q}}. \tag{4.55}$$

Since $\widehat{S}_{\hat{q}}$ is invertible, $B(s)$ is uniquely specified by $\widehat{\mathrm{vec}}_{\mathrm{t}} B$. □

A less general version of Theorem 4.8 can be derived if only d_B, the degree of $B(s)$, is considered instead of each entry degree. Matrix $\widehat{S}(s)$ is then replaced by $S(s)$.

Corollary 4.5 *Given a non-negative integer d_B and*

$$\begin{aligned} &\bullet\ q = \tfrac{1}{2}n^2(d_B+1) && \textit{if } d_B \textit{ is odd} \\ &\bullet\ q = \tfrac{1}{2}n(n+1) + \tfrac{1}{2}n^2 d_B && \textit{otherwise} \end{aligned} \tag{4.56}$$

interpolation triplets (s_k, a_k, b_k) for $k = 1, 2, \ldots, q$ such that the q-by-q matrix

$$S_q = \begin{bmatrix} a_1 S(s_1) \\ a_2 S(s_2) \\ \vdots \\ a_q S(s_q) \end{bmatrix}$$

is non-singular, there exists a unique n-by-n para-Hermitian matrix $B(s)$ of degree d_B for which (4.54) holds for $k = 1, 2, \ldots, q$.

Proof See Theorem 4.8. The number of interpolation points can be obtained by inspection of the number of columns of matrix $S(s)$ in (4.51) and clearly verifies $q \geq \hat{q}$. That is, the lack of information on entry degrees of $B(s)$ is somehow compensated by a greater number of interpolation points. $\square$

Example 4.4 *For the matrix $B(s)$ given in Example 4.2, $d_B = 2$ and*

$$\Delta = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}.$$

Theorem 4.8 therefore needs $\hat{q} = 5$ interpolation points, while Corollary 4.5 makes use of $q = 7$ points to uniquely characterize $B(s)$. These values correspond to respective column dimensions of matrices $\widehat{S}(s)$ and $S(s)$ in Example 4.3.

4.5.3 Interpolation formulation of the symmetric matrix polynomial equation

Kronecker product and permutation matrices are first used to derive an alternative expression for the symmetric matrix polynomial equation. An equivalence is then established between solutions $X(s)$ to (4.48) of expected column degrees and solutions to a linear system of equations derived by interpolation.

Let $A(s) = A_0 + sA_1 + \ldots + s^{d_A} A_{d_A}$ and $X(s) = X_0 + sX_1 + \ldots + s^{d_X} X_{d_X}$. Recalling Section 3.3, the Kronecker product formulation of (4.48) leads to

$$[\mathbf{I}_n \otimes A^T(-s)] \,\mathrm{vec}X(s) + [A^T(s) \otimes \mathbf{I}_n] \,\mathrm{vec}X^T(-s) = 2\, vecB(s). \tag{4.57}$$

With

$$Q(s) = \mathbf{I}_n \otimes A^T(s) = Q_0 + sQ_1 + \ldots + s^{d_A} Q_{d_A}$$

equation (4.57) becomes

$$Q(-s)\,\mathrm{vec}X(s) + PQ(s)\,\mathrm{vec}X(-s) = 2\,\mathrm{vec}B(s). \tag{4.58}$$

In addition to that, suppose that matrices $A(s)$ and $X(s)$ have respective column degrees d_{iA} and d_{iX} for $i = 1, 2, \ldots, n$. With the notation

$$X_j = [x_1^{(j)}\; x_2^{(j)}\; \cdots\; x_n^{(j)}]$$

to distinguish between columns of matrix coefficient X_j for $j = 0, 1, \ldots, d_X$, it is useful to note that for $i = 1, 2, \ldots, n$ and $j > d_{iX}$

$$x_i^{(j)} = \mathbf{0} \tag{4.59}$$

and then that corresponding entries of $\mathrm{vec}X_j$ are zero.

Finally, define $\text{vec}X$ as the constant column vector obtained by stacking together vectors $\text{vec}X_j$ for $j = 0, 1, \ldots, d_X$.

By regrouping ascending powers of s of the right hand-side, polynomial equation (4.58) can be written

$$L(s)\,\text{vec}X = 2\,\text{vec}B(s) \tag{4.60}$$

$$L(s) = [L_{+1}(s) \quad sL_{-1}(s) \quad s^2L_{+1}(s) \ \ldots] \tag{4.61}$$

and

$$\begin{array}{rcl} L_{+1}(s) & = & Q(-s) + PQ(s) \\ L_{-1}(s) & = & Q(-s) - PQ(s). \end{array}$$

Since components of $\text{vec}X$ corresponding to columns (4.59) are zero, a more compact formulation of (4.60) can be written. For that, define $\widehat{\text{vec}}X$ as the reduced column vector obtained by stacking for $j = 0, 1, \ldots, d_X$ columns $x_i^{(j)}$ of X_j for which $j \leq d_{iX}$. That is to say, $\widehat{\text{vec}}X$ is obtained from $\text{vec}X$ by suppressing zero components corresponding to columns (4.59).

Rewriting of (4.60) follows from appropriate reduction of $L(s)$ into a matrix $\widehat{L}(s)$ of compatible column dimension :

$$L(s)\,\text{vec}X = \widehat{L}(s)\,\widehat{\text{vec}}X = 2\,\text{vec}B(s). \tag{4.62}$$

Bilateral equation (4.48) is thus equivalent to unilateral equation (4.62). The polynomial unknown $X(s)$ is replaced by the constant vector unknown $\text{vec}X$ or $\widehat{\text{vec}}X$ containing coefficients of $X(s)$.

4.5.4 Interpolation results

If now $\widetilde{B}(s)$ is a para-Hermitian matrix such that

$$2\,\text{vec}\widetilde{B}(s) = \widehat{L}(s)\,\widehat{\text{vec}}X \tag{4.63}$$

then by inspection, degree entries of $\widetilde{B}(s)$ verify for $i, j = 1, 2, \ldots, n$

$$\begin{array}{l} \bullet\ d_{ii\tilde{B}} = d_{iA} + d_{iX} \ \text{ if } d_{iA} + d_{iX} \text{ is even} \\ \bullet\ d_{ii\tilde{B}} = d_{iA} + d_{iX} - 1 \ \text{ if } d_{iA} + d_{iX} \text{ is odd} \\ \bullet\ d_{ij\tilde{B}} = \max(d_{iA} + d_{jX}, d_{jA} + d_{iX}) \text{ when } i \neq j. \end{array} \tag{4.64}$$

Now according to Theorem 4.8, $\widetilde{B}(s)$ is uniquely specified by

$$\hat{q} = \frac{1}{2}n(n+1) + \frac{1}{2}\sum_{i,j=1}^{n} d_{ij\tilde{B}} \tag{4.65}$$

interpolation points. If triplets (s_k, a_k, b_k) for $k = 1, 2, \ldots, \hat{q}$ are chosen as in Theorem 4.8 and such that

$$a_k \mathrm{vec}\widetilde{B}(s_k) = a_k \mathrm{vec}B(s_k) = b_k \tag{4.66}$$

then to set $s = s_k$ and to pre-multiply (4.60) by a_k leads to

$$\widehat{L}_{\hat{q}}\ \widehat{\mathrm{vec}}X = 2\ T_{\hat{q}} \tag{4.67}$$

where

$$\widehat{L}_{\hat{q}} = \begin{bmatrix} a_1\widehat{L}(s_1) \\ a_2\widehat{L}(s_2) \\ \vdots \\ a_{\hat{q}}\widehat{L}(s_{\hat{q}}) \end{bmatrix} \tag{4.68}$$

has dimensions $\hat{q}$-by-$n(n + \sum_{i=1}^{n} d_{iX})$ and $T_{\hat{q}}$ is defined in (4.55).

We are now in position to formulate our main result.

Theorem 4.9 *Given $A(s)$ and $B(s)$ in (4.48), select non-negative integers d_{iX} to satisfy for $i, j = 1, 2, \ldots, n$*

$$d_{ijB} \leq \max(d_{iA} + d_{jX}, d_{jA} + d_{iX}) \tag{4.69}$$

where d_{iA} are column degrees of $A(s)$ and d_{ijB} entry degrees of $B(s)$. Moreover, suppose that linear system (4.67) is built from interpolation triplets verifying the assumptions of Theorem 4.8. Then a solution $X(s)$ to (4.48) of (highest) column degrees d_{iX} exists if and only if a solution $\widehat{\mathrm{vec}}X$ to linear system (4.67) exists.

Proof If a solution to (4.48) of column degrees d_{iX} exists, then clearly a solution to (4.67) exists. In view of (4.64), inequality (4.69) is also necessary.

Conversely, if a solution to (4.67) exists then according to (4.55) and (4.66),

$$\widehat{L}_{\hat{q}}\ \widehat{\mathrm{vec}}X = 2\ \widehat{S}_{\hat{q}}\ \widehat{\mathrm{vec}}_{\mathfrak{t}}\widetilde{B} = 2\ \widehat{S}_{\hat{q}}\ \widehat{\mathrm{vec}}_{\mathfrak{t}}B$$

which implies $\widetilde{B}(s) = B(s)$ since $\widehat{S}_{\hat{q}}$ is non-singular and $\widehat{\mathrm{vec}}_{\mathfrak{t}}B$ uniquely characterizes $B(s)$ in Theorem 4.8. In view of (4.63), polynomial equation (4.58) admits a solution $X(s)$ with column degrees d_{iX}. □

Theorem 4.9 shows that there exists a one-to-one correspondence between matrix polynomial solutions to (4.48) with given column degrees and constant vector solutions to linear system of equations (4.67). Therefore to apply numerical methods to (4.67) provides a way to characterize all solutions of expected column degrees to the symmetric matrix polynomial equation.

Note however that Theorem 4.9 does not specify how to find the interpolation triplets required to built linear system (4.67) or even guarantee the existence of a solution to (4.48). These points will be considered later on.

Furthermore, it is important to recall that up to now, no assumptions were made concerning polynomial matrices $A(s)$ and $B(s)$.

A simplified result can be derived if now $d_{iX} = d_X$ and $d_{iA} = d_A$ for $i = 1, 2, \ldots, n$. Inequality (4.69) then reads

$$d_{ijB} \leq d_B \leq d_A + d_X \tag{4.70}$$

which is clearly necessary for a polynomial solution $X(s)$ to (4.48) to exist. Now consider Theorem 4.9 with $\widehat{L}(s) = L(s)$ and $\widehat{\mathrm{vec}}X = \mathrm{vec}X$. According to Corollary 4.5, para-Hermitian matrix $B(s)$ is interpolated using q points and one obtains the linear system

$$L_q \, \mathrm{vec}X = 2\, T_q \tag{4.71}$$

where L_q is q-by-$n^2(d_X + 1)$. Since now only the knowledge of d_B is required to select d_X, the following corollary to Theorem 4.9 has been shown.

Corollary 4.6 *Given $A(s)$ and $B(s)$ in (4.48), select non-negative integer d_X to satisfy (4.70). Suppose moreover that linear system (4.71) is built from interpolation triplets verifying the assumptions of Corollary 4.5. Then a solution $X(s)$ of (highest) degree d_X exists if and only if a solution* $\mathrm{vec}X$ *to linear system (4.71) exists.*

Note that Corollary 4.6 needs more interpolation points than Theorem 4.9 and thus that linear system (4.71) is of greater size than linear system (4.67).

Although this is out of the scope of this paper, it is worthwhile to keep in mind that interpolation also provides useful flexibility properties. As pointed out in [2], a crucial advantage of interpolation over other approaches is that input matrices need not be fixed. In other words, using interpolation one could solve problems where $B(s)$ is not necessarily fixed in equation (4.48) but is known, say, through the zeros of its determinant. This remarkable feature is specific to interpolation.

4.5.5 Reduced resultant formulation of the symmetric matrix polynomial equation

Algorithms in [46] are based on the fact that equations involving polynomials can be solved by equating coefficients of equal powers of the indeterminate s, or equivalently that the product of two polynomial matrices can be formulated via multiplication of two constant matrices. One of them is called the Sylvester resultant matrix [11, 22]. Results concerning unilateral polynomial equations involving matrix product can therefore be derived from a corresponding constant matrix formulation.

The present section aims to show that such a technique can also be applied to bilateral equations. After some preliminaries, an equivalence between polynomial solutions to (4.48) and constant solutions to a linear system based on a resultant matrix is stated.

In particular it is shown that the dimensions of the original problem can be significantly reduced. In addition to provide an alternative way to solve (4.48), links with stability theory and Hurwitz matrices are enlightened. Relations with the interpolation method are drawn and both techniques are shown to be closely related. Moreover the resultant approach turns out to be useful when combined with interpolation results. In particular it is possible, under some assumptions on $A(s)$, to give a priori conditions of existence of solutions.

Preliminaries: projectors and related matrices

Recall that P is the n^2-by-n^2 permutation matrix. Then

$$P_{+1} = (\mathbf{I}_{n^2} + P)/2$$

is called projector and

$$P_{-1} = \mathbf{I}_{n^2} - P_{+1} = (\mathbf{I}_{n^2} - P)/2$$

is its complementary projector. By simple inspection one can check that

$$\begin{array}{rcl} \text{rank } P_{+1} & = & n(n+1)/2 \\ \text{rank } P_{-1} & = & n(n-1)/2. \end{array}$$

Hence let us define

- $\widetilde{\Pi}_{+1}$ as a matrix with $n(n+1)/2$ rows corresponding to the linearly independent rows of P_{+1}
- $\widetilde{\Pi}_{-1}$ as a matrix with $n(n-1)/2$ rows corresponding to the linearly independent rows of P_{-1}

and

$$\begin{array}{rcl} \Pi_{+1} & = & \left[\begin{array}{c} \widetilde{\Pi}_{+1} \\ \mathbf{0} \end{array} \right] \\ \Pi_{-1} & = & \left[\begin{array}{c} \mathbf{0} \\ \widetilde{\Pi}_{-1} \end{array} \right] \end{array}$$

as square matrices of size n^2. According to properties (4.50), for k even

$$\begin{gathered} \widetilde{\Pi}_{+1} \mathrm{vec} B_k = \mathrm{vec}_{\mathrm{t}} B_k \\ \widetilde{\Pi}_{-1} \mathrm{vec} B_{k+1} = \mathrm{vec}_{\mathrm{t}} B_{k+1} \\ [\Pi_{+1} \quad \Pi_{-1}] \left[\begin{array}{c} \mathrm{vec} B_k \\ \mathrm{vec} B_{k+1} \end{array} \right] = \mathrm{vec}_{\mathrm{t}} (B_k + s B_{k+1}) \end{gathered} \tag{4.72}$$

and

$$\begin{aligned} \widetilde{\Pi}_{-1}\mathrm{vec}B_k &= \mathbf{0} \\ \widetilde{\Pi}_{+1}\mathrm{vec}B_{k+1} &= \mathbf{0} \\ [\Pi_{-1} \quad \Pi_{+1}]\begin{bmatrix} \mathrm{vec}B_k \\ \mathrm{vec}B_{k+1} \end{bmatrix} &= \mathbf{0}. \end{aligned} \tag{4.73}$$

Note finally that

$$\mathrm{rank}\begin{bmatrix} \widetilde{\Pi}_{+1} \\ \widetilde{\Pi}_{-1} \end{bmatrix} = n^2 \tag{4.74}$$

and whereas Π_{+1} and Π_{-1} are both singular, that

$$\mathrm{rank}\,[\Pi_{+1} \quad \Pi_{-1}] = \mathrm{rank}\begin{bmatrix} \Pi_{+1} \\ \Pi_{-1} \end{bmatrix} = n^2. \tag{4.75}$$

Reduced resultant formulation

Suppose that inequality (4.70) holds, i.e. $d_X \geq d_B - d_A \geq 0$. This is a natural hypothesis since only polynomial solutions to (4.48) are expected. Finally without loss of generality, assume that $d_B = d_A + d_X$. If $d_B < d_A + d_X$ simply note $B(s) = B(s) + B_{d_B+1}s^{d_B+1} + \cdots + B_{d_B+\delta}s^{d_B+\delta}$ where $\delta = d_A + d_X - d_B$ and B_{d_B+i} for $i = 1, 2, \ldots, \delta$ are zero matrices. Then let $d_B = d_B + \delta$.

In Section 3.2 the Kronecker product formulation of (4.48) has been shown to lead to equation (4.58) :

$$Q(-s)\,\mathrm{vec}X(s) + PQ(s)\,\mathrm{vec}X(-s) = 2\,\mathrm{vec}B(s).$$

The Sylvester resultant formulation is derived by equating coefficients of the powers of s in the above matrix polynomial equation :

$$T\begin{bmatrix} \mathrm{vec}X \\ \mathrm{vec}X \end{bmatrix} = 2\mathrm{vec}B \tag{4.76}$$

where

$$T = \left[\begin{array}{cccc|cccc} Q_0 & & & \mathbf{0} & PQ_0 & & & \mathbf{0} \\ -Q_1 & Q_0 & & & PQ_1 & -PQ_0 & & \\ \vdots & -Q_1 & \ddots & Q_0 & \vdots & -PQ_1 & \ddots & (-1)^{d_X}PQ_0 \\ (-1)^{d_A}Q_{d_A} & \vdots & & -Q_1 & PQ_{d_A} & \vdots & & (-1)^{d_X}PQ_1 \\ & (-1)^{d_A}Q_{d_A} & \ddots & \vdots & & -PQ_{d_A} & \ddots & \vdots \\ \mathbf{0} & & & (-1)^{d_A}Q_{d_A} & \mathbf{0} & & & (-1)^{d_X}PQ_{d_A} \end{array}\right] \tag{4.77}$$

is an $n^2(d_A + d_X + 1)$-by-$2n^2(d_X + 1)$ constant matrix. Pre- and post-multiplication by the following matrices transform T into block-diagonal form :

$$\frac{1}{2}\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} T \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix} = \begin{bmatrix} R_{d_X} & \mathbf{0} \\ \mathbf{0} & R^*_{d_X} \end{bmatrix} \tag{4.78}$$

with if $m = d_A + d_X + 1$ is even

$$\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{\frac{m}{2}} \otimes [\Pi_{+1} \quad \Pi_{-1}] \\ \mathbf{I}_{\frac{m}{2}} \otimes [\Pi_{-1} \quad \Pi_{+1}] \end{bmatrix} \tag{4.79}$$

and otherwise

$$\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = \left[\begin{array}{cc} \mathbf{I}_{\frac{m-1}{2}} \otimes [\Pi_{+1} \quad \Pi_{-1}] & \mathbf{0} \\ \mathbf{0} & \widetilde{\Pi}_{+1} \\ \hline \mathbf{I}_{\frac{m-1}{2}} \otimes [\Pi_{-1} \quad \Pi_{+1}] & \mathbf{0} \\ \mathbf{0} & \widetilde{\Pi}_{-1} \end{array}\right]. \tag{4.80}$$

Properties (4.74) and (4.75) show that (4.79) and (4.80) are invertible matrices and therefore that (4.78) is an equivalence relation. Note that matrix R_{d_X} has row dimension $n^2m/2$ when m is even and $n^2(m-1)/2 + n(n+1)/2$ otherwise. After substitution of (4.78) into (4.76), one obtains

$$\begin{bmatrix} R_{d_X} \operatorname{vec} X \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} M_1 \operatorname{vec} B \\ M_2 \operatorname{vec} B \end{bmatrix}. \tag{4.81}$$

According to equation (4.72) the upper part of the right hand-side matrix in (4.81) is $\operatorname{vec}_{\mathrm{t}} B$ and the lower part is readily seen to be zero by property (4.73). Therefore (4.81) is equivalent to

$$R_{d_X} \operatorname{vec} X = \operatorname{vec}_{\mathrm{t}} B, \tag{4.82}$$

hence leading to a significant reduction of original problem dimensions.

Now suppose that matrix $X(s)$ has column degrees d_{iX} for $i = 1, 2, \ldots, n$ and that $\widehat{\operatorname{vec}} X$ is obtained from $\operatorname{vec} X$ by suppressing zero components associated to columns (4.59). Moreover, suppose that in a similar fashion, corresponding columns of R_{d_X} are left out to yield a matrix $\widehat{R}_{d_X}$ such that

$$\widehat{R}_{d_X} \widehat{\operatorname{vec}} X = \operatorname{vec}_{\mathrm{t}} B. \tag{4.83}$$

The equivalence between polynomial matrix solutions to (4.48) and constant vector solutions to (4.82) and (4.83) is summarized by the following statements, which are respective alternatives to Theorem 4.9 and Corollary 4.6.

Theorem 4.10 *Given $A(s)$ and $B(s)$ in (4.48), select non negative integers $d_{iX} \leq d_X$ for $i = 1, 2, \ldots, n$ such that d_X satisfy (4.70). Then a solution $X(s)$ to (4.48) of column degrees d_{iX} exists if and only if a solution $\widehat{\operatorname{vec}} X$ to linear system (4.83) exists.*

Corollary 4.7 *Given $A(s)$ and $B(s)$ in (4.48), select non negative integer d_X to satisfy (4.70). Then a solution $X(s)$ to (4.48) of degree d_X exists if and only if a solution* $\operatorname{vec} X$ *to linear system (4.82) exists.*

Relation with stability theory and Hurwitz matrices. Rank properties of the reduced resultant

A particularly interesting case which deserves a mention is when

$$d_X = d_A - 1 \tag{4.84}$$

and

$$\begin{array}{l} \bullet\ A(s) \text{ is regular, i.e. } A_{d_A} \text{ is non-singular} \\ \bullet\ \text{determinants of } A(s) \text{ and } A(-s) \text{ are relatively prime.} \end{array} \tag{4.85}$$

Lemma 4.9 *Under assumptions (4.84) and (4.85), the reduced resultant matrix R_{d_X} in (4.82) is non-singular.*

Proof Matrix R_{d_X} in (4.82) is square and built from a Sylvester resultant matrix in (4.77) which is made of four square Toeplitz block-matrices of striped form :

$$T = \left[\begin{array}{cccc|cccc} Q_0 & & & \mathbf{0} & PQ_0 & & & \mathbf{0} \\ -Q_1 & Q_0 & & & PQ_1 & -PQ_0 & & \\ \vdots & & \ddots & & \vdots & & \ddots & \\ (-1)^{m-1}Q_{m-1} & \cdots & & Q_0 & PQ_{m-1} & \cdots & & (-1)^{m-1}PQ_0 \\ \hline (-1)^m Q_m & \cdots & & -Q_1 & PQ_m & \cdots & & (-1)^m PQ_1 \\ & (-1)^m Q_m & & Q_2 & & -PQ_m & & (-1)^m PQ_2 \\ & & \ddots & \vdots & & & \ddots & \vdots \\ \mathbf{0} & & & (-1)^m Q_m & \mathbf{0} & & & (-1)^m PQ_m \end{array}\right].$$

Theorem 4.26 in [4] states that for $A(s)$ regular the above matrix is non-singular if and only if the determinants of $A(s)$ and $A(-s)$ are relatively prime. Furthermore, in view of equivalence relation (4.78), T non-singular implies R_{d_X} non-singular. □

In the scalar case ($n = 1$) R_{d_X} can be directly related to the Hurwitz matrix of polynomial $A(s)$ [40], which is known to be non-singular when $A(s)$ does not have symmetric roots with respect to the imaginary axis. A simple example should help to clarify that.

Example 4.5 *Consider the symmetric polynomial equation ($n = 1$)*

$$a(-s)x(s) + x(-s)a(s) = 2b(s)$$

where $a(s) = a_0 + a_1 s + a_2 s^2 + a_3 s^3$, $x(s) = x_0 + x_1 s + x_2 s^2$ and $b(s) = b_0 + b_2 s^2 + b_4 s^4$. Note that assumption (4.84) is verified and therefore that R_{d_X} will be square. Using the Sylvester resultant formulation, (4.76) reads

$$\left[\begin{array}{ccc|ccc} a_0 & 0 & 0 & a_0 & 0 & 0 \\ -a_1 & a_0 & 0 & a_1 & -a_0 & 0 \\ a_2 & -a_1 & a_0 & a_2 & -a_1 & a_0 \\ \hline -a_3 & a_2 & -a_1 & a_3 & -a_2 & a_1 \\ 0 & -a_3 & a_2 & 0 & -a_3 & a_2 \\ 0 & 0 & -a_3 & 0 & 0 & a_3 \end{array}\right] \left[\begin{array}{c} x_0 \\ x_1 \\ x_2 \\ x_0 \\ x_1 \\ x_2 \end{array}\right] = 2 \left[\begin{array}{c} b_0 \\ 0 \\ b_2 \\ 0 \\ b_4 \\ 0 \end{array}\right]$$

which after reduction becomes (4.82)

$$R_2 \,\mathrm{vec} X = \begin{bmatrix} a_0 & 0 & 0 \\ a_2 & -a_1 & a_0 \\ 0 & -a_3 & a_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_2 \\ b_4 \end{bmatrix} = \mathrm{vec}_{\mathrm{t}} B.$$

The Hurwitz matrix of $a(s)$ defined in [40] is

$$H(a) = \begin{bmatrix} a_2 & a_0 & 0 \\ a_3 & a_1 & 0 \\ 0 & a_2 & a_0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} R_2^T \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

and therefore $a(s)$ strictly stable and regular ($a_3 \neq 0$) implies $H(a)$ and R_2 non-singular.

The following extension of Lemma 4.9 will be useful when combined with interpolation results.

Lemma 4.10 *Under assumptions (4.85) and if*

$$d_X \geq d_A - 1 \tag{4.86}$$

the reduced resultant matrix R_{d_X} in (4.82) has full row-rank.

Proof The reduction process of Theorem 4.10 leads to a reduced resultant of the form

$$R_{d_X} = \left[\begin{array}{c|c} R_S & R_C \\ \hline \mathbf{0} & R_R \end{array}\right]$$

where R_S is the non-singular reduced resultant corresponding to $d_X = d_A - 1$ (see Lemma 4.9) and R_R is a matrix whose rows are obtained after reduction of two block triangular matrices with respectively Q_{d_A} and PQ_{d_A} on their diagonals. By assumption (4.85), Q_{d_A} is non-singular and therefore R_R has full row-rank. $\square$

Note that Lemma 4.10 is a strengthened version of Lemma 4.9 since equality (4.84) is only a particular case of inequality (4.86).

Comparison with interpolation formulation

It is stimulating to understand relations existing between linear system (4.71) :

$$L_q \,\mathrm{vec} X = 2\, T_q$$

obtained in the scope of interpolation, and linear system (4.82) :

$$R_{d_X} \,\mathrm{vec} X = \mathrm{vec}_{\mathrm{t}} B$$

derived from the Sylvester resultant theory.

By construction one can check that matrix $L(s)$ given in (4.62) verifies

$$L(s) = 2\, S(s)\, R_{d_X} \tag{4.87}$$

where $S(s)$ was defined in (4.51) and R_{d_X} is the reduced resultant of Corollary 4.7. Pre-multiplication of (4.87) by a_j at points s_j for $j = 1, 2, \ldots, q$ yields

$$L_q = 2\, S_q\, R_{d_X} \tag{4.88}$$

Since $S_q \operatorname{vec_t} B = T_q$ one can conclude that linear system (4.71) is simply linear system (4.82) multiplied to the left by non-singular matrix $2\, S_q$.

Existence of solution

It is often of interest to know beforehand whether a solution to (4.48) can be found or not. The following condition is established in this regard.

Lemma 4.11 *Given $A(s)$ and $B(s)$ in (4.48) where $A(s)$ verifies assumptions (4.85), select d_X such that (4.86) holds. Then a solution $X(s)$ of (highest) degree d_X to (4.48) exists and is given by linear system of equation (4.71).*

Proof Consider Corollary 4.6 where equivalence between existence of a polynomial solution to (4.48) and existence of a constant solution to linear system (4.71) was established.

Observe that a solution $\operatorname{vec} X$ to (4.71) exists if and only if

$$\operatorname{rank}\, [L_q \quad T_q] = \operatorname{rank}\, L_q. \tag{4.89}$$

A sufficient condition for (4.89) to hold is that L_q has full row-rank. Otherwise, some relations between L_q and T_q (that is between $A(s)$ and $B(s)$) must exist. Since S_q is invertible, equation (4.88) yields

$$\operatorname{rank}\, L_q = \operatorname{rank}\, R_{d_X}. \tag{4.90}$$

According to Lemma 4.10 the corresponding reduced resultant R_{d_X} has full row-rank. Equation (4.90) shows that this is also the case for L_q and therefore that (4.71) always has a solution. $\square$

4.5.6 Numerical methods

It is now possible to expose numerical methods to solve (4.48). Even if our main impetus is to expose an interpolation technique to solve the symmetric bilateral matrix polynomial equation, the Sylvester matrix results previously developed will first be shown to supply an alternative resolution method. Two interpolation algorithms are then described. The choice of interpolation points, a crucial point, is discussed at length. By the use of simple examples these methods are shown to be easily extended to handle cases like presence of purely imaginary zeros or special requirements on the expected solution. Finally, comparative numerical results are exposed that clearly demonstrate the efficiency and reliability of our algorithms over already existing techniques.

Reduced resultant algorithm

According to Theorem 4.10, there is a correspondence between polynomial solutions to (4.48) and constant vector solutions to the reduced linear system (4.83) obtained by equating the coefficients of s. This leads to the subsequent straightforward formulation of a first algorithm to solve (4.48).

Algorithm SymSyl

1. Given $A(s)$, $B(s)$ in (4.48), select d_{iX} for $i = 1, 2, \ldots, n$ to be expected column degrees of solution $X(s)$.
2. Using relation (4.78), build matrix $\widehat{R}_{d_X}$ and linear system (4.83).
3. Each solution $\widehat{\text{vec}}X$ to linear system (4.83) corresponds to a solution $X(s)$ to (4.48) of column degrees d_{iX}.

Interpolation algorithm

In contrast to Algorithm SymSyl, the following procedures are now based on interpolation and are implementations of Theorem 4.9 and related linear system (4.67).

Algorithm SymInt

1. Given $A(s)$, $B(s)$ in (4.48), select d_{iX} for $i = 1, 2, \ldots, n$ to be expected column degrees of solution $X(s)$. They must verify (4.69).
2. Compute $d_{ij\tilde{B}}$ using (4.64), and then the number of interpolation points $\hat{q}$ according to (4.65).
3. Build $\widehat{L}(s)$ by selecting columns of $L(s)$ in (4.61) corresponding to d_{iX}.

4. Select distinct scalars s_i and random vectors a_i for $i = 1, 2, \ldots, \hat{q}$ to build $\widehat{L}_{\hat{q}}$ in (4.68) and $T_{\hat{q}}$ in (4.55).

5. Each solution $\widehat{\text{vec}}X$ to linear system (4.67) corresponds to a solution $X(s)$ to (4.48) of column degrees d_{iX}.

Theorem 4.9 states that in order to build linear system (4.67), interpolation points must verify the assumptions of Theorem 4.8, that is, matrix $\widehat{S}_{\hat{q}}$ in (4.53) must be non-singular. Lemma 2.4 in [2] shows that for a choice of $\hat{q}$ distinct scalars s_i, $\widehat{S}_{\hat{q}}$ is non-singular for almost any set of vectors a_i, thus justifying the use of random vectors in step (4) of the above algorithm.

However, the special structure of $\widehat{S}(s)$ suggests a more selective choice for vectors a_i and scalars s_i. In that sense, the modification to Algorithm SymInt described below allows to build linear system (4.67) given a restricted number, say n_e, of scalars t_i verifying for $i, j = 1, 2, \ldots, n_e$ and $i \neq j$

$$|t_i| \neq |t_j| \neq 0 \tag{4.91}$$

and with a relatively simple set of vectors a_i such that matrix $\widehat{S}_{\hat{q}}$ is guaranteed to be non-singular. Therefore Theorem 4.9 can also be invoked.

Algorithm SymInt' Only step (4) differs from Algorithm SymInt:

4. Construct matrix $\Delta^* = [d^*_{ijB}]$ where

$$\begin{array}{ll} \bullet\ d^*_{ijB} = \frac{1}{2}d_{ij\tilde{B}} + 1 & \text{if } d_{ij\tilde{B}} \text{ is even and } i \geq j \\ \bullet\ d^*_{ijB} = \frac{1}{2}d_{ij\tilde{B}} & \text{if } d_{ij\tilde{B}} \text{ is even and } i < j \\ \bullet\ d^*_{ijB} = \frac{1}{2}(d_{ij\tilde{B}} + 1) & \text{if } d_{ij\tilde{B}} \text{ is odd.} \end{array} \tag{4.92}$$

   Choose $n_e = \max\{d^*_{ijB} : i, j = 1, 2, \ldots, n\}$ distinct scalars t_i verifying (4.91). Then for $i = 1, 2, \ldots, n_e$

   (a) set $s = t_i$ in (4.62) to obtain the linear system

$$\widehat{L}(t_i)\widehat{\text{vec}}X = 2\,\text{vec}B(t_i) \tag{4.93}$$

   (b) for $k = 1, 2, \ldots, n^2$, if the k-th component of vecΔ^* is greater than or equal to i, the k-th row of linear system (4.93) is selected to build linear system (4.67).

Row selection step (4b) corresponds, for a fixed $s = t_i$, to choose a set a_j of unitary vectors. Since polynomial equations (4.48) and (4.62) are equivalent, the above procedure amounts

- to evaluate the symmetric matrix polynomial equation at n_e interpolation points t_i verifying assumption (4.91)
- to interpolate a solution $X(s)$ by selecting a sufficient amount of data in terms of rows of equation (4.93) when i varies from 1 to n_e.

Note that the total number of rows selected is $\sum_{i,j=1}^{n} d^*_{ijB} = \hat{q}$.

Furthermore, matrix $\widehat{S}_{\hat{q}}$ in (4.53) can be checked (after some row and column permutations) to be a block diagonal matrix compound of Vandermonde matrices which, for choice (4.91) of scalars t_i, are known to be invertible, as shown by the simple following example.

Example 4.6 *Consider the strictly Hurwitz polynomial matrix*

$$A(s) = \begin{bmatrix} s^2+3s+1 & s+1 \\ -1 & 2s^2+4s \end{bmatrix}$$

and the para-Hermitian polynomial matrix

$$B(s) = \begin{bmatrix} s^2+1 & \frac{1}{2}(s^3-2s^2-s+1) \\ \frac{1}{2}(-s^3-2s^2+s+1) & s^2+1 \end{bmatrix}.$$

First of all we study solutions $X(s)$ to (4.48) with column degrees $d_{1X}=0$ and $d_{2X}=1$. Inequalities (4.69) are verified and polynomial matrix $\widetilde{B}(s)$ in (4.63) has entry degrees

$$\begin{aligned} \tilde{d}_{11} &= d_{1X}+d_{1A} = 2 \\ \tilde{d}_{12} &= \max(d_{1X}+d_{2A}, d_{2X}+d_{1A}) = 3 \\ \tilde{d}_{21} &= \tilde{d}_{12} = 3 \\ \tilde{d}_{22} &= d_{2X}+d_{2A}-1 = 2. \end{aligned} \tag{4.94}$$

The number of points required to interpolate $\widetilde{B}(s)$ in Theorem 4.8 is then $\hat{q} = 3+(2+3+3+2)/2 = 8$. With $d_{1X}=0$ and $d_{2X}=1$, polynomial matrix $\widehat{L}(s)$ in (4.62) is built on $L_{+1}(s)$ and only the two last columns of $sL_{-1}(s)$ to give

$$\widehat{L}(s) = \left[\begin{array}{cccc|cc} 2s^2+2 & -2 & 0 & 0 & 0 & 0 \\ -s+1 & 2s^2-4s & s^2+3s+1 & -1 & -s^3-3s^2-s & s \\ s+1 & 2s^2+4s & s^2-3s+1 & -1 & s^3-3s^2+s & -s \\ 0 & 0 & 2 & 4s^2 & -2s^2 & -8s^2 \end{array}\right].$$

Using equalities (4.92), one obtains $d^*_{ijB}=2$ for $i,j=1,2$ and clearly $n_e=2$. Since all elements in matrix Δ^* are the same, to set $s=t_1=1$ and $s=t_2=2$ into equation (4.62) provides an easy way to write (4.67)

$$\begin{bmatrix} \widehat{L}(1) \\ \widehat{L}(2) \end{bmatrix} \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} 2\,\mathrm{vec}B(1) \\ 2\,\mathrm{vec}B(2) \end{bmatrix}. \tag{4.95}$$

The matrix $\widehat{S}_{\hat{q}}$ in (4.53) corresponding to entry degrees (4.94) reads

$$\widehat{S}_{\hat{q}} = \begin{bmatrix} \widehat{S}(1) \\ \widehat{S}(2) \end{bmatrix}$$

and after some row and column permutations, $\widehat{S}_{\hat{q}}$ appears to be equivalent to

$$\begin{bmatrix} 1 & t_1^2 & & & & & & \\ 1 & t_2^2 & & & & & & \\ & & 1 & t_1^2 & & & & \\ & & 1 & t_2^2 & & & & \\ & & & & 1 & t_1 & t_1^2 & t_1^3 \\ & & & & 1 & -t_1 & t_1^2 & -t_1^3 \\ & & & & 1 & t_2 & t_2^2 & t_2^3 \\ & & & & 1 & -t_2 & t_2^2 & -t_2^3 \end{bmatrix}.$$

For choice (4.91) of scalars t_i, that is $|t_1| \neq 0$, $|t_2| \neq 0$, $|t_1| \neq |t_2|$, this block Vandermonde matrix is non-singular and Theorem 4.9 can be applied. Linear system (4.95) is composed of 8 equations for 6 scalar unknowns and admits a (unique) solution $\widehat{\mathrm{vec}} X = [1\ 0\ 1\ 1\ 1\ 0]^T$, i.e.

$$X(s) = \begin{bmatrix} 1 & s+1 \\ 0 & 1 \end{bmatrix} \tag{4.96}$$

is the unique solution to (4.48) with column degrees $d_{1X} = 0$ and $d_{2X} = 1$.

If d_{1X} is now expected to be 1, the values of $d_{ij\tilde{B}}$ in (4.94) are not modified in view of (4.64). The characterization of all solutions $X(s)$ to (4.48) is then direct, since $A(s)$ verify assumptions (4.85), inequality (4.86) holds and the number of interpolation points given by Corollary 4.11 is still 8. L_q is known by advance to have full row-rank and is checked to be square. The unique solution to (4.48) of degree $d_X = 1$ is therefore still (4.96).

Critical case

Since Theorem 4.9 does not make any assumption on $A(s)$ and $B(s)$, it can also handle the case when $A(s)$ has purely imaginary zeros. Clearly Lemma 4.11 cannot be applied anymore but equation (4.67) still provides a way to conclude on the existence of a solution.

Example 4.7 *Let*

$$A(s) = \begin{bmatrix} s+1 & 1 \\ s & 0 \end{bmatrix}$$

and

$$B(s) = \begin{bmatrix} -s^2+1 & \frac{1}{2}(-s+1) \\ \frac{1}{2}(s+1) & 0 \end{bmatrix}$$

polynomial matrices for which all solutions $X(s)$ to (4.48) with degree $d_X = 1$ have to be found. Note that $A(s)$ has a (purely imaginary) zero at $s = 0$. Inequalities (4.69) are verified and (4.64) yields $d_{11\tilde{B}} = 2$, $d_{12\tilde{B}} = 2$ and $d_{22\tilde{B}} = 0$ which set the number of interpolation points to $\hat{q} = 3 + (2 + 2 + 2 + 0)/2 = 6$. Simple polynomial matrix *manipulations show that*

$$L_{+1}(s) = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 0 & s+1 & s \\ 1 & 0 & -s+1 & -s \\ 0 & 0 & 2 & 0 \end{bmatrix}$$

and

$$L_{-1}(s) = \begin{bmatrix} -2s & -2s & 0 & 0 \\ 1 & 0 & -s-1 & -s \\ -1 & 0 & -s+1 & -s \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Equations (4.92) give

$$\Delta^* = \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}$$

and $n_e = 2$. *To set for instance* $s = t_1 = 1$ *and* $s = t_2 = 2$ *in (4.60) provides* 8 *linear constraints on* $\widehat{\mathrm{vec}}X = \mathrm{vec}X$ *from which (according to step (4b) in Algorithm* SYMINT*') only* 6 *are selected to build* $\widehat{L}_{\hat{q}} = L_{\hat{q}}$ *and* $T_{\hat{q}}$. *Equation (4.67) then reads*

$$\left[\begin{array}{cccc|cccc} 2 & 0 & 0 & 0 & -2 & -2 & 0 & 0 \\ 1 & 0 & 2 & 1 & 1 & 0 & -2 & -1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ \hline 2 & 0 & 0 & 0 & -8 & -8 & 0 & 0 \\ 1 & 0 & 3 & 2 & 2 & 0 & -6 & -4 \end{array}\right] \begin{bmatrix} \mathrm{vec}X_0 \\ \mathrm{vec}X_1 \end{bmatrix} = \left[\begin{array}{c} 0 \\ 2 \\ 0 \\ 0 \\ \hline -6 \\ 3 \end{array}\right].$$

It can be verified that

$$\mathrm{rank}\, L_{\hat{q}} = \mathrm{rank}\, [L_{\hat{q}} \quad T_{\hat{q}}] = 5. \tag{4.97}$$

Using standard numerical tools the following parameterization of all solutions $X(s)$ *of degree* $d_X = 1$ *to (4.48) is obtained :*

$$X(s) = \begin{bmatrix} (1-a)s+1 & -bs \\ as+c & bs+a-b \end{bmatrix}$$

where a, b *and* c *are any scalars.*

Additional constraints

It is worthwhile to note that when in system (4.67) there are more unknowns than equations, i.e

$$n(n + \sum_{i=1}^{n} d_{iX}) > \hat{q},$$

this freedom can be handled so that $X(s)$ satisfies supplementary constraints. Linear system (4.67) can then be augmented of $n(n + \sum_{i=1}^{n} d_{iX}) - \hat{q}$ additional linear equations on elements of $\widehat{\mathrm{vec}}X$, that is on scalar coefficients of $X(s)$:

$$\begin{bmatrix} \widehat{L}_{\hat{q}} \\ D \end{bmatrix} \widehat{\mathrm{vec}}X = \begin{bmatrix} 2T_{\hat{q}} \\ E \end{bmatrix}.$$

Example 4.8 *To force $X(s)$ in Example 4.7 to be for instance upper triangular is equivalent to add to system (4.67) the two following linear constraints :*

$$\left[\begin{array}{cccc|cccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \mathrm{vec} X_0 \\ \mathrm{vec} X_1 \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \end{array}\right].$$

All upper triangular solutions of degree $d_X = 1$ are then

$$X(s) = \left[\begin{array}{cc} s+1 & -ds \\ 0 & ds \end{array}\right]$$

where d is any scalar. The presence of a parameter is not surprising since the original $L_{\hat{q}}$ does not have full row-rank, as shown by equation (4.97).

4.5.7 Computer simulations

Computer simulations were conducted to evaluate the performance of our methods in comparison with that of [19].

We implemented and compared four numerical routines :

- The elementary polynomial operations algorithm based on Euclidean division, proposed in [19].
- Algorithm SymSyl, resolution of a linear system built on a reduced Sylvester resultant matrix.
- Algorithm SymInt, resolution of a linear system obtained from random interpolation points.
- Algorithm SymInt', resolution of a linear system obtained from specific interpolation points.

Some comments regarding implementation are in order.

- Even though performance of Algorithms SymSyl, SymInt and SymInt' mainly rely upon the linear system solver being used, we have here simply invoked the Matlab built-in Gaussian elimination algorithm.
- Crucial steps in the Euclidean division algorithm are the computation of an adjoint polynomial matrix and the extraction of a polynomial matrix left factor (see equations (11) and (12) in [19]). Theoretically, both tasks can be achieved using triangularization via elementary polynomial operations. Practically, numerical results were so poor that we had to implement Sylvester resultant methods instead [46]. Elementary polynomial operations are thus only used to solve via Euclidean division scalar versions of polynomial equation (4.48), that is equation (14) in [19].

For our computer simulations, $A(s)$ and $X(s)$ were chosen to be random square polynomial matrices of size n. For convenience, we supposed that degrees of both matrices were the same, i.e. $d_A = d_X$. Each constant scalar coefficient of polynomial entries of $A(s)$ and $X(s)$ was a Gaussian random variable with zero mean and unitary variance, except for the strictly lower triangular entries of $X(s)$ that were set to zero. Matrix $B(s)$ is evaluated from $A(s)$ and $X(s)$ using equation (4.48).

Input matrices $A(s)$ and $B(s)$ being given, the performance of each equation solver was measured by the root mean square error, defined as

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_r}\sum_{k=1}^{N_r}\gamma(\hat{X}_k(s) - X(s))^2}$$

where N_r is the number of independent runs (100 in our case), $\hat{X}_k(s)$ denotes the solution computed in the k-th run, $X(s)$ is the exact solution to equation (4.48) and finally

$$\gamma(X(s)) = \sqrt{(\mathrm{vec}X)^T \mathrm{vec}X}.$$

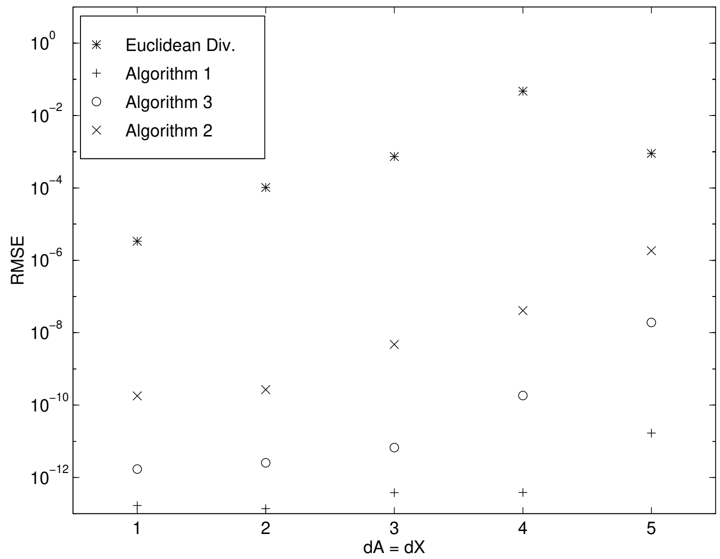

Figure 4.1: Comparative performance for $n = 3$.

Figure 4.1 shows the RMSEs of the four methods as functions of $d_A = d_X$. The following comments naturally arise.

- One can see that our routines are more accurate and reliable than the Euclidean division algorithm. Generally speaking, there exist well-worked standard solvers to deal with linear systems of equations (even more powerful than the one used for our simulations). On the contrary, elementary polynomial operations can behave erratically.

- Interpolation Algorithms SymInt and SymInt' are less accurate than reduced resultant Algorithm SymSyl. We can even notice than the selective choice of Algorithm SymInt' seems numerically preferable to the random selection of interpolation points in Algorithm SymInt. Actually, the choice of interpolation points is known to affect the condition number of matrix $\widehat{L}_{\hat{q}}$ in linear system (4.67). A random choice typically guarantees an adequate conditioning. However, alternative (other than $[1\, s\, s^2 \ldots]$) polynomial basis such as Chebychev polynomials and corresponding interpolation points can improve the condition number many times, see Section 3 in [2]. This possibility has not been exploited in the present paper.

4.6 Discrete-time symmetric equation

4.6.1 Introduction

We consider the discrete-time symmetric matrix polynomial equation

$$A^T(\mathtt{d}^{-1})X(\mathtt{d}) + X^T(\mathtt{d}^{-1})A(\mathtt{d}) = B(\mathtt{d}) \tag{4.98}$$

where $A(\mathtt{d})$ is a given polynomial matrix with real coefficients, $B(\mathtt{d})$ is a given two-sided para-Hermitian polynomial matrix with real coefficients such that

$$\begin{array}{rcl} B(\mathtt{d}) & = & B_l(\mathtt{d}^{-1}) + B_r(\mathtt{d}) \\ B_l(\mathtt{d}) & = & B_r^T(\mathtt{d}) \end{array} \tag{4.99}$$

and $X(\mathtt{d})$ is an n-by-n polynomial matrix to be found. This bilateral equation in polynomial matrices arises in various areas of system control theory and signal processing.

Its probably most famous application resides in the discrete-time polynomial matrix spectral factorization, a fundamental step in the algebraic approach to control and filtering problems such as LQ optimal regulation, Kalman filtering or minimum variance LQG control [31]. In [19], a spectral factorization algorithm based on a Newton-Raphson scheme consisting of equation (4.98) at every iteration was proposed.

Another significant role played by this equation stems from signal processing. In [45] it is shown that equation (4.98) may be used to compute the covariance matrix function of a multivariable ARMA process, a problem frequently encountered in estimation, filtering, stochastic control and communications.

In spite of the relevance of equation (4.98), only a few works have been devoted so far to its resolution by means of systematic numerical procedures. In [19], the authors used an algorithm based on polynomial reductions that has been studied in further detail in [20]. It relies upon elementary polynomial operations and can be considered as an extension of the Euclidean division algorithm for polynomials. Although these basic algebraic operations are suitable for educational and theoretical purposes, their lack of good numerical properties becomes problematic when attempting a computer implementation. The experience reveals that when working with a matrix-oriented software like MATLAB, a more efficient alternative is required. A different approach has been followed in [45] where it is shown on a simple example that solving (4.98) is actually equivalent to solving a linear system of equations. However, the authors only considered the case of scalar polynomials ($n = 1$) and did not tackle further the problem of designing a systematic numerical procedure for solving (4.98).

Based on the above considerations, our main contribution is a numerically reliable algorithm that solves (4.98). Moreover, critical cases such as the presence of zeros of $\mathbf{A}(\mathtt{d})$ on the unit circle can be dealt with in a straightforward way. That was not the case with the algorithm proposed in [19, 20].

The section is organized as follows. An equivalent constant matrix formulation is first obtained by equating coefficients of powers of $\mathtt{d}$ in (4.98). This technique is usually referred to as the Sylvester matrix approach to solving polynomial equations. Moreover, symmetry properties (4.99) can be taken into account to significantly reduce the dimensions of the equivalent linear system of equations. Applying on this system standard and well-worked numerical routines then guarantees the algorithm robustness. Then, we show that under some assumptions on $A(\mathtt{d})$, existence as well as uniqueness of solution to (4.98) can be guaranteed. Three simple numerical examples borrowed from control and signal processing literature underline both simplicity and efficiency of this new numerical method. Finally, numerical advantages of our algorithm over previously published methods are illustrated by intensive computational experiments.

Note that, in contrast to the continuous-time case, we do not propose an interpolation algorithm for solving (4.98). However, this should be done without extra difficulty.

4.6.2 Reduced Sylvester matrix formulation

In this section, we aim to show that matrix polynomial equation (4.98) can be solved without performing any polynomial operations. For that purpose, we will establish a bijective correspondence between every solution to (4.98) and a constant vector solution to a linear system of equations which order can be kept reasonable thanks to symmetry properties (4.99).

First, equation (4.98) can be written in terms of Kronecker product and vectors (see Section 3.3)

$$[\mathbf{I}_n \otimes A^T(\mathtt{d}^{-1})]\mathrm{vec}X(\mathtt{d}) + [A^T(\mathtt{d}) \otimes \mathbf{I}_n]\mathrm{vec}X^T(\mathtt{d}^{-1}) = \mathrm{vec}B(\mathtt{d}) \tag{4.100}$$

where $\mathrm{vec}X(\mathtt{d}) = \mathrm{vec}X_0 + \mathrm{vec}X_1\mathtt{d} + \cdots + \mathrm{vec}X_{d_X}\mathtt{d}^{d_X}$ is a polynomial vector obtained by stacking together columns of polynomial matrix $X(\mathtt{d})$. As in the continuous-time case (see the previous section), we shall make use of the square permutation matrix P and its related projectors $P_+, P_-, \tilde{\Pi}_+$ and $\tilde{\Pi}_-$. Moreover, we define the non-singular matrix

$$\Pi = \begin{bmatrix} \tilde{\Pi}_+ \\ \tilde{\Pi}_- \end{bmatrix} \tag{4.101}$$

According to commutativity properties of Kronecker product [40], equation (4.100) yields

$$Q(\mathtt{d}^{-1})\mathrm{vec}X(\mathtt{d}) + PQ(\mathtt{d})\mathrm{vec}X(\mathtt{d}^{-1}) = \mathrm{vec}B_r(\mathtt{d}) + P\mathrm{vec}B_r(\mathtt{d}^{-1}) \tag{4.102}$$

for

$$Q(\mathtt{d}) = \mathbf{I}_n \otimes A^T(\mathtt{d}) = Q_0 + Q_1\mathtt{d} + \cdots Q_{d_A}\mathtt{d}^{d_A}.$$

Let us denote the degrees of $A(\mathtt{d})$, $B_r(\mathtt{d})$ and $X(\mathtt{d})$ by d_A, d_B and d_X, respectively. Assume without loss of generality that $\max(d_X, d_A) = \deg B_r(\mathtt{d}) = d_B$. This assumption is not restrictive since for $X(\mathtt{d})$ to be polynomial, inequality $\max(d_X, d_A) - \deg B_r(\mathtt{d}) = \delta \geq 0$ must hold. Then if $\delta > 0$, simply note $B_r(\mathtt{d}) = B_r(\mathtt{d}) + \sum_{i=\deg B_r(\mathtt{d})}^{d_B} B_{r\,i}\mathtt{d}^i$ where the $B_{r\,i}$ are zero matrices and let $\deg B_r(\mathtt{d}) = \deg B_r(\mathtt{d}) + \delta = d_B$.

With these conventions and the results recalled in the previous section, the Sylvester matrix formulation of matrix polynomial equation (4.102) reads

$$\mathcal{A}\underbrace{\begin{bmatrix} \mathrm{vec}X_0 \\ \mathrm{vec}X_1 \\ \vdots \\ \mathrm{vec}X_{d_X} \end{bmatrix}}_{\mathrm{vec}X} + \tilde{\mathcal{A}}\begin{bmatrix} \mathrm{vec}X_{d_X} \\ \vdots \\ \mathrm{vec}X_1 \\ \mathrm{vec}X_0 \end{bmatrix} = \begin{bmatrix} P\mathrm{vec}B_{r\,d_B} \\ \vdots \\ P\mathrm{vec}B_{r\,1} \\ P_+\mathrm{vec}B_{r\,0} \\ \mathrm{vec}B_{r\,1} \\ \vdots \\ \mathrm{vec}B_{r\,d_B} \end{bmatrix} = \begin{bmatrix} \mathcal{B} \\ P_+\mathrm{vec}B_{r\,0} \\ \mathcal{B}^* \end{bmatrix} \tag{4.103}$$

where

$$\mathcal{A} = \begin{bmatrix} < \mathtt{d}^{d_A}Q(\mathtt{d}^{-1}) >_{d_X} \\ \mathbf{0}_{n^2(2d_B - d_A - d_X)\times n^2(d_X+1)} \end{bmatrix} \tag{4.104}$$

and

$$\tilde{\mathcal{A}} = \begin{bmatrix} \mathbf{0}_{n^2(2d_B - d_A - d_X)\times n^2(d_X+1)} \\ < PQ(\mathtt{d}) >_{d_X} \end{bmatrix}. \tag{4.105}$$

Note now that if $\mathbf{J}_k$ is a k-by-k matrix having ones along the anti-diagonal and zeros elsewhere, we can write (4.103) such that unknown scalar coefficients of $X(\mathtt{d})$ appear only in the vector $\mathrm{vec}X$:

$$T\begin{bmatrix} \mathrm{vec}X \\ \mathrm{vec}X \end{bmatrix} = \begin{bmatrix} \mathcal{B} \\ P_+\mathrm{vec}B_{r\,0} \\ \mathcal{B}^* \end{bmatrix} \tag{4.106}$$

for

$$T = [\mathcal{A} \quad \tilde{\mathcal{A}}(\mathbf{J}_{d_X} \otimes \mathbf{I}_{n^2})]. \tag{4.107}$$

By simple inspection, it can be shown that pre- and post-multiplication by the following non-singular matrices transform T into block diagonal form :

$$\frac{1}{2}\begin{bmatrix} \mathbf{I}_{n^2 d_B} & 0 & \mathbf{J}_{d_B}\otimes P \\ 0 & \Pi & 0 \\ -\mathbf{I}_{n^2 d_B} & 0 & \mathbf{J}_{d_B}\otimes P \end{bmatrix} T \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathcal{Q} & \\ \tilde{P}_+\mathcal{Q} & 0 \\ & \tilde{P}_-\mathcal{Q} \\ 0 & -\mathcal{Q} \end{bmatrix} = \begin{bmatrix} \mathcal{Q}_R & 0 \\ 0 & \mathcal{Q}_R^* \end{bmatrix} \tag{4.108}$$

where $\mathcal{Q} = [Q_0\, Q_1\, \cdots\, Q_{d_X}]$ and $Q_k = \mathbf{0}$ if $k > d_A$.

Since $\mathcal{B} = (\mathbf{J}_{d_B}\otimes P)\mathcal{B}^*$, substitution of (4.108) into (4.106) gives

$$\begin{bmatrix} \mathcal{Q}_R & 0 \\ 0 & \mathcal{Q}_R^* \end{bmatrix}\begin{bmatrix} \mathrm{vec}X \\ 0 \end{bmatrix} = \begin{bmatrix} \mathcal{B} \\ \tilde{P}_+\mathrm{vec}B_{r0} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathcal{B}_R \\ 0 \end{bmatrix} \tag{4.109}$$

which in turn yields

$$\mathcal{Q}_R \mathrm{vec}X = \mathcal{B}_R \tag{4.110}$$

by considering the non-zero rows in (4.109). Note that $\mathcal{Q}_R$ has dimensions $n^2 d_B + n(n+1)/2$ and $n^2(d_X+1)$. Because (4.108) is an equivalence relation involving non-singular matrices, there exists a one-to-one correspondence between every solution $X(\mathtt{d})$ to matrix polynomial equation (4.98) and the constant vector solution $\mathrm{vec}X$ of linear system of equations (4.110). The following theorem has been shown.

Theorem 4.11 *Equation (4.98) has a polynomial solution $X(\mathtt{d})$ if and only if there exists a constant vector solution* $\mathrm{vec}X$ *to the reduced linear system of equation (4.110).*

Algorithm DSymSyl

Input	Polynomial matrices $A(\mathtt{d})$, $B(\mathtt{d})$ verifying symmetry properties (4.99).
Output	Polynomial matrix $X(\mathtt{d})$ solution to matrix polynomial equation (4.98).
Step 1	Build matrix T in (4.107) from relations (4.104) and (4.105).
Step 2	Using equations (4.101) and (4.108), extract the reduced Sylvester matrix $\mathcal{Q}_R$.
Step 3	Build matrix $\mathcal{B}_R$ according to equations (4.103) and (4.109).
Step 4	Every constant vector solution $\mathrm{vec}X$ to linear system (4.110) corresponds to a polynomial matrix solution $X(\mathtt{d})$ to matrix polynomial equation (4.98).

Previously published methods dealing with the symmetric matrix polynomial equation made use of elementary polynomial operations featuring bad numerical behavior. The technique exposed here only relies upon the resolution of a linear system of equations, for which powerful and reliable tools are known to exist [15]. Furthermore, symmetry properties (4.99) have been considered so as to significantly reduce the original problem dimensions.

4.6.3 Existence and shape of solution

Since we dispose now of an efficient method to solve equation (4.98), it is natural to wonder whether it has a solution or not. In this optic, Ježek [20] has shown the following useful result.

Theorem 4.12 *Suppose that matrix $A(\mathtt{d})$ is stable in the discrete-time sense, i.e.* $\det A(\mathtt{d}) \neq 0$ *for $|\mathtt{d}| \leq 1$. Then matrix polynomial equation (4.98) has a solution. Furthermore, if $A(0)$ is upper triangular and non-singular, there exists a* unique *solution $X(\mathtt{d})$ to (4.98) with $X(0)$ upper triangular.*

The proof of Theorem 4.12 provided in [20] is based on elementary polynomial operations, but the results can be translated in terms of properties verified by linear system (4.110) in the following way.

Corollary 4.8 *If $A(\mathtt{d})$ is stable in the discrete-time sense, then*

$$\mathrm{rank}[\mathcal{Q}_R \quad \mathcal{B}_R] = \mathrm{rank}\,\mathcal{Q}_R. \tag{4.111}$$

Moreover if $A(0)$ is upper triangular, then after insertion of $n(n-1)/2$ supplementary linear equations setting the strictly lower part of $X(0)$ to zero, linear system (4.110) reads

$$\begin{bmatrix} \mathcal{Q}_R \\ E \end{bmatrix} \mathrm{vec}X = \mathcal{Q}_T \mathrm{vec}X = \begin{bmatrix} \mathcal{B}_R \\ \mathbf{0} \end{bmatrix} = \mathcal{B}_T \tag{4.112}$$

where $\mathcal{Q}_T$ is a non-singular matrix of order $n^2(d_B+1)$.

Proof Assuming $\deg X(\mathtt{d}) = \deg B_r(\mathtt{d}) = d_B$, rank condition (4.111) and non-singularity of matrix $\mathcal{Q}_T$ in (4.112) are immediate consequences of existence and uniqueness conditions of Theorem 4.12. □

The numerical aspect of Corollary 4.8 is especially interesting since the existence of a solution to linear system (4.110) is guaranteed *a priori*. That is, no preliminary rank tests have to be performed on matrices $\mathcal{Q}_R$ and $\mathcal{B}_R$.

4.6.4 Applications

Spectral factorization

First consider the problem of extracting a stable factor from the para-Hermitian matrix

$$B(\mathtt{d}) = \begin{bmatrix} -4\mathtt{d}^{-1} + 18 - 4\mathtt{d} & -2\mathtt{d}^{-1} - 11 \\ -11 & -5\mathtt{d}^{-1} + 35 - 5\mathtt{d} \end{bmatrix}$$

with zeros in 0.10,0.50,2 and 10. That is to say, we seek a matrix $A(\mathtt{d})$ such that

$$\begin{array}{l} A^T(\mathtt{d}^{-1})A(\mathtt{d}) = B(\mathtt{d}) \\ \det A(\mathtt{d}) \neq 0 \text{ for } |\mathtt{d}| \leq 1. \end{array}$$

According to [19], the Newton-Raphson iterative process

- $A_k^T(\mathtt{d}^{-1})X_k(\mathtt{d}) + X_k^T(\mathtt{d}^{-1})A_k(\mathtt{d}) = B(\mathtt{d})$
  for upper triangular $X_k(0)$ (4.113)
- $A_{k+1}(\mathtt{d}) = \frac{1}{2}(A_k(\mathtt{d}) + X_k(\mathtt{d}))$ (4.114)

for $k = 0, 1, 2, \ldots$ and a non-singular stable initial matrix $A_0(\mathtt{d})$ such that $A_0(0)$ is upper triangular, converges quadratically to a stable matrix $A(\mathtt{d}) = A_\infty(\mathtt{d})$ solution to the spectral factorization problem. Following [19], we choose $A_0(\mathtt{d}) = \Psi$ as the Cholesky square root matrix of the diagonal leading coefficient of $B(\mathtt{d})$, viz. $\Psi^T\Psi = B(0)$. We will now show how equation (4.113) can be solved at a given step k, using linear system (4.112) and Corollary 4.8.

After $k = 2$ iterations of process (4.113-4.114), we obtain

$$A_2(\mathtt{d}) = \begin{bmatrix} 4.0189 - 0.9925\mathtt{d} & -2.9528 \\ -0.9658\mathtt{d} & 5.0506 - 0.9875\mathtt{d} \end{bmatrix}$$

and we seek a solution $X_2(\mathtt{d})$ to equation (4.113). Equivalent linear system (4.112) is easily obtained from equations (4.107) and (4.108). We have

$$\underbrace{\left[\begin{array}{rrrrrrrr} -0.99 & -0.97 & 0 & 0 & 4.02 & 0 & 0 & 0 \\ 0 & -0.99 & 0 & 0 & 0 & 0 & 4.02 & 0 \\ 0 & 0 & -0.99 & -0.97 & -2.95 & 5.05 & 0 & 0 \\ 0 & 0 & 0 & -0.99 & 0 & 0 & -2.95 & 5.05 \\ 16.08 & 0 & 0 & 0 & -3.97 & -3.86 & 0 & 0 \\ -5.91 & 10.10 & 8.04 & 0 & 0 & -1.98 & -1.99 & -1.93 \\ 0 & 0 & -11.81 & 20.20 & 0 & 0 & 0 & -3.95 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]}_{\mathcal{Q}_T} \mathrm{vec}X_2 = \underbrace{\left[\begin{array}{r} -8 \\ 0 \\ -4 \\ -10 \\ 72 \\ -44 \\ 140 \\ \hline 0 \end{array}\right]}_{\mathcal{B}_T}.$$

Note the last row in $\mathcal{Q}_T$ and $\mathcal{R}_T$ to force the absolute coefficient matrix of the solution to be upper triangular. Corollary 4.8 assumptions hold and therefore $\mathcal{Q}_T$ is non-singular. The unique solution $\mathrm{vec}X_2$ to linear system (4.112) yields

$$X_2(\mathtt{d}) = \begin{bmatrix} 3.9820 - 1.0072\mathtt{d} & -3.0453 \\ -1.0324\mathtt{d} & 4.9516 - 1.0118\mathtt{d} \end{bmatrix}$$

that allows to build $A_3(\mathtt{d})$, see equation (4.114).

Repeating this process with a working precision of 12 decimals, after 5 iterations we eventually get

$$A_5(\mathtt{d}) = \begin{bmatrix} 4 - \mathtt{d} & -3 \\ -\mathtt{d} & 5 - \mathtt{d} \end{bmatrix}$$

which appears to be the desired spectral factor. Its zeros are indeed checked to be 2 and 10.

Spectral factorization: critical case

Even when $A(\mathtt{d})$ does not verify the assumptions of Corollary 4.8, linear system (4.112) still provides a way to solve equation (4.98). However, preliminary rank tests are now mandatory to conclude on the existence of a solution.

To see that, let be given

$$A(\mathtt{d}) = \begin{bmatrix} \mathtt{d} & 1 \\ 1 & -1+\mathtt{d}+\mathtt{d}^2 \end{bmatrix}$$

having only critical zeros located in -1,-1 and 1. First select

$$B_1(\mathtt{d}) = \begin{bmatrix} 2 & 2+2\mathtt{d} \\ 2\mathtt{d}^{-1}+2 & \mathtt{d}^{-2}+2\mathtt{d}^{-1}+2\mathtt{d}+\mathtt{d}^2 \end{bmatrix}$$

as the right hand side matrix in equation (4.98). Equivalent linear system (4.110) reads

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -2 & 0 & 2 & 0 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & -4 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 4 \end{bmatrix} \mathrm{vec}X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 2 \\ 0 \\ 2 \\ 4 \\ 4 \\ 0 \end{bmatrix}.$$

Using basic numerical linear algebra tools, we solve the above system and find the following parameterization of all polynomial solutions $X(\mathtt{d})$ to equation (4.98) :

$$X(\mathtt{d}) = \begin{bmatrix} -2+\mathtt{d} & 1+\mathtt{d} \\ 2\mathtt{d} & 1 \end{bmatrix} + a(\mathtt{d}) \begin{bmatrix} 1 & -1+\mathtt{d}+\mathtt{d}^2 \\ -\mathtt{d} & -1 \end{bmatrix} + b(\mathtt{d}) \begin{bmatrix} 1 & -1 \\ -\mathtt{d} & \mathtt{d} \end{bmatrix} + c(\mathtt{d}) \begin{bmatrix} -1+\mathtt{d} & -2\mathtt{d} \\ -1+\mathtt{d} & 1+\mathtt{d}^2 \end{bmatrix}$$

for $a(\mathtt{d})$,$b(\mathtt{d})$ and $c(\mathtt{d})$ arbitrary scalar polynomials. In the same way, linear system (4.112) gives a parameterization of all polynomial solutions $X(\mathtt{d})$ having an upper triangular absolute coefficient matrix X_0. To see that, just set $c(\mathtt{d}) = 0$ in the previous expression.

If we choose at present

$$B_2(\mathtt{d}) = \begin{bmatrix} 1 & 2+2\mathtt{d} \\ 2\mathtt{d}^{-1}+2 & \mathtt{d}^{-2}+2\mathtt{d}^{-1}+2\mathtt{d}+\mathtt{d}^2 \end{bmatrix}$$

as the right hand side matrix in equation (4.98), inspection of corresponding linear system (4.110) shows that $\mathrm{rank}\mathcal{Q}_R = 9$ while $\mathrm{rank}[\mathcal{Q}_R \quad \mathcal{B}_R] = 10$ and therefore that (4.98) does not admit any polynomial solution in this case.

Computing the covariance function of an ARMA process

Consider the real ARMA process

$$A(\mathtt{d})y_k = B(\mathtt{d})e_k$$

where k is a positive integer and $\mathtt{d}$ is the forward shift operator defined by $\mathtt{d}y_k = y_{k+1}$. $A(\mathtt{d})$ and $B(\mathtt{d})$ are square polynomial matrices with real coefficients. The random sequence e is a white noise so that

$$E[e_k e_l^T] = \begin{cases} 0 \text{ for } k \neq l \\ I \text{ for } k = l. \end{cases}$$

Polynomial matrix $A(\mathtt{d})$ is assumed to have all its roots strictly inside the unit circle. Under these assumptions the ARMA process y is well-defined and asymptotically stationary. The problem we are to solve is finding the covariance matrix function

$$R(\tau) = \lim_{k\to\infty} E[y_{k+\tau} y_k^T].$$

It can be computed by inverse z-transformation of the spectral density matrix

$$\Phi(\mathtt{d}) = [A(\mathtt{d})]^{-1}C(\mathtt{d})C^T(\mathtt{d}^{-1})[A^T(\mathtt{d}^{-1})]^{-1}.$$

The computation follows [45] by partial fraction expansion of the spectral density in the form

$$\Phi(\mathtt{d}) = [A(\mathtt{d})]^{-1}X(\mathtt{d}) + X^T(\mathtt{d}^{-1})[A^T(\mathtt{d}^{-1})]^{-1}.$$

This is equivalent to solving the symmetric matrix polynomial equation

$$X(\mathtt{d})A^T(\mathtt{d}^{-1}) + A(\mathtt{d})X^T(\mathtt{d}^{-1}) = C(\mathtt{d})C^T(\mathtt{d}^{-1}) \tag{4.115}$$

for the polynomial matrix $X(\mathtt{d})$. Once X is available, we may expand

$$[A(\mathtt{d})]^{-1}X(\mathtt{d}) = \sum_{\tau=0}^{\infty} \hat{R}(\tau)\mathtt{d}^{-\tau}$$

by long polynomial division. Inspection shows that

$$R(\tau) = \begin{cases} \hat{R}(\tau) & \text{for } \tau > 0 \\ \hat{R}(0) + \hat{R}^T(0) & \text{for } \tau = 0 \\ \hat{R}^T(-\tau) & \text{for } \tau < 0. \end{cases}$$

To illustrate the procedure, consider a two-variable ARMA process y defined by

$$A(\mathtt{d}) = \begin{bmatrix} 1-2\mathtt{d} & 0 \\ 6 & 1-2.5\mathtt{d} \end{bmatrix} \quad C(\mathtt{d}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Matrix polynomial equation (4.115) is actually a transposed version of (4.98), so that after performing the necessary transpositions our algorithm yields the linear system of equations

$$\begin{bmatrix} -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -2.5 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 0 & 6 & 1 & 0 & 0 \\ 0 & 0 & 0 & -2.5 & 0 & 0 & 6 & 1 \\ 2 & 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 6 & 1 & 1 & 0 & 0 & -2.5 & -2 & 0 \\ 0 & 0 & 12 & 2 & 0 & 0 & 0 & -5 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathrm{vec} X_T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Solution vector $\mathrm{vec}X_T = [-0.167 \;\, 0 \;\, -1.5 \;\, -1.81 \;\, -0.333 \;\, -1 \;\, 0 \;\, -4.52]^T$ provides a polynomial matrix solution to (4.115), namely

$$X(\mathtt{d}) = \begin{bmatrix} -0.167 - 0.333\mathtt{d} & -\mathtt{d} \\ -1.5 & -1.81 - 4.52\mathtt{d} \end{bmatrix}.$$

The desired covariance function is recovered by long division of computed polynomial matrix $X(\mathtt{d})$ by given polynomial matrix $A(\mathtt{d})$. The first N values of the function may be obtained through

$$[A(\mathtt{d})]^{-1}X(\mathtt{d})\mathtt{d}^N = R(N) + R(N-1)\mathtt{d} + \cdots + R(0)\mathtt{d}^N + [A(\mathtt{d})]^{-1}Y(\mathtt{d})$$

where $[A(\mathtt{d})]^{-1}Y(\mathtt{d})$ is strictly proper. We have used the matrix polynomial division macro of the Polynomial Toolbox [P10] to get the covariance matrix function represented in Figure 1.

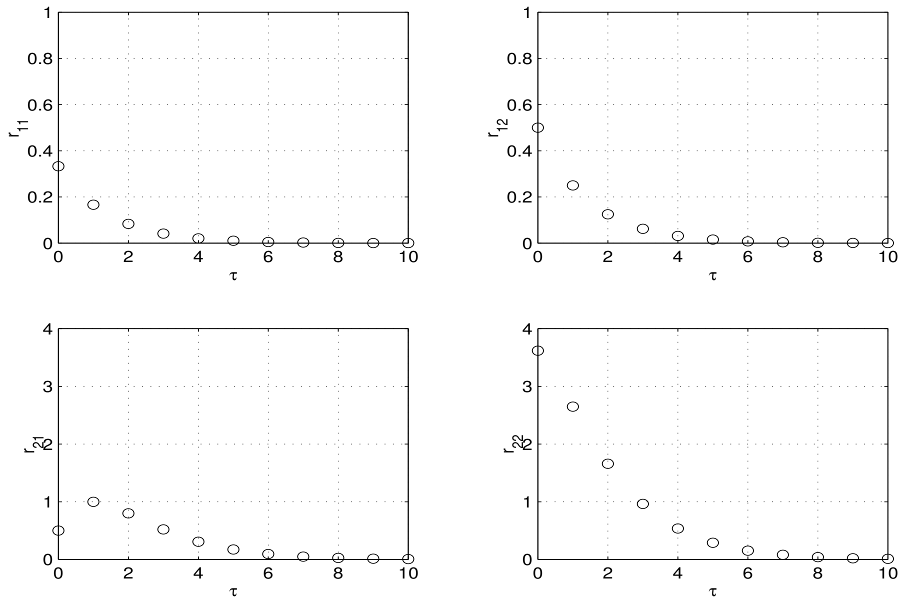

Figure 1: ARMA Process Covariance Matrix Function

4.6.5 Numerical experiments

Computer simulations were conducted to evaluate the performance of our method in comparison with already existing methods. Some comments regarding implementation are in order.

- The Sylvester matrix algorithm (presented here) is based on Sylvester matrices. We simply invoked the MATLAB built-in Gaussian elimination algorithm to solve linear system (4.110).
- The polynomial reduction algorithm (presented in [19]) is based on extended Euclidean division. Crucial steps are the computation of an adjoint polynomial matrix and the extraction of a polynomial matrix left factor, see equations (11) and (12) in [19]. Theoretically, both tasks can be achieved using triangularization via elementary polynomial operations. Practically, numerical results were so poor that we had to implement our own Sylvester resultant methods instead. Elementary polynomial operations were thus only used to solve via Euclidean division scalar versions of polynomial equation (4.98), that is equation (14) in [19].

For our computer simulations, $A(\mathtt{d})$ and $X(\mathtt{d})$ were chosen to be random square polynomial matrices of size n. For convenience, we supposed that degrees of both matrices were the same, i.e. $d_A = \deg A(\mathtt{d}) = \deg X(\mathtt{d}) = d_X$. Each constant scalar coefficient of polynomial entries of $A(\mathtt{d})$ and $X(\mathtt{d})$ was a Gaussian random variable with mean zero and variance two, except for the strictly lower triangular entries of $X(\mathtt{d})$ that were set to zero. Matrix $B(\mathtt{d})$ is evaluated from $A(\mathtt{d})$ and $X(\mathtt{d})$ using equation (4.98).

Input matrices $A(\mathtt{d})$ and $B(\mathtt{d})$ being given, each equation solver returns a solution $\hat{X}_k(\mathtt{d})$ that can be compared with the known exact solution $X(\mathtt{d})$ to equation (4.98). The default tolerance used for solving linear systems and polynomial equations has been set to 10^{-8}. If the resolution fails when performing elementary operations or solving linear systems of equations with this tolerance, then the whole equation solver is considered to have failed. For each fixed value of n and $d = d_A = d_X$, 100 independent runs of each equation solver were performed. The percentage of failures for each algorithm is reported in Table 1. The average execution times with MATLAB 5.1 on a 300 Mhz Sun SparcStation 30 are reported in Table 2.

Sylvester matrix algorithm

n / d	1	2	3	4	5
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

Polynomial reduction algorithm

n / d	1	2	3	4	5
2	0	0	1	0	0
3	0	2	2	9	18
4	13	64	99	100	100

Table 1: Percentage of failures.

Sylvester matrix algorithm

n / d	1	2	3	4	5
2	0.10	0.11	0.13	0.14	0.16
3	0.12	0.14	0.17	0.20	0.23
4	0.13	0.17	0.22	0.29	0.38

Polynomial reduction algorithm

n / d	1	2	3	4	5
2	0.40	0.45	0.50	0.56	0.63
3	0.61	0.74	0.88	1.1	1.3
4	0.81	1.04	1.4	–	–

Table 2: Execution times in seconds.

One can see that our algorithm is more reliable than the Euclidean division algorithm. Generally speaking, there exist well-worked standard solvers to deal with linear systems of equations (even more powerful than the one used for our simulations). On the contrary, elementary polynomial operations can behave erratically. Moreover, linear system resolution is definitively faster than elementary polynomial operations. Indeed, MATLAB is rather a matrix-oriented software whereas performing operations on polynomials usually requires a substantial amount of time-consuming nested loops.

Chapter 5

Závěr (Conclusion)

The thesis first features a comprehensive survey of linear matrix polynomial equations arising in control and existing numerical methods for solving them. The lack of sufficiently general, reliable and efficient algorithms is underlined. The remaining part of our work is then devoted to bridge the gap between existing algorithms dealing with polynomial matrices and efficient numerical linear algebra routines involving stable operations. It results in tractable new algorithms.

A distinguishing feature of our approach is the employment of Sylvester matrices and interpolation, two popular and well-understood polynomial techniques. Our algorithms are designed for overcoming numerical instability, a problem which has plagued for long years control engineers using old fashioned numerical methods for solving matrix polynomial equations. In addition to that, new insights were lent into relevant polynomial matrix problems:

- We provided two methods for evaluating the rank of a polynomial matrix that are based on constant matrix rank computation. In view of the availability of powerful tools that deal with constant matrices, it is believed that these new algorithms constitute a significant improvement over existing elementary operation techniques. Links with polynomial null-space extraction were also enlightened. More specifically, an upper bound on the degree of the null-space of an arbitrary polynomial matrix was established. An application to a highly general type of polynomial matrix equations arising in control problems was finally mentioned in terms of a numerically tractable condition of existence of a solution.
- Our triangularization algorithms may also be used for polynomial rank evaluation, polynomial null-space extraction or greatest common divisor computation. From the point of view of numerical stability, the advantage of using these new polynomial triangularization techniques over using the standard Euclidean division algorithm may be compared to the advantage of using orthogonal reductions over using the standard Gaussian elimination algorithm.
- We proposed two interpolation formulations of continuous-time para-Hermitian ma-

trices. The first one was used together with the Kronecker product to characterize all solutions of expected column degrees to the bilateral symmetric equation via a linear system of equations of reduced size. The development of an alternative resolution method, this time based on Sylvester resultant matrices, was also motivated for theoretical reasons since some relations with stability theory and Hurwitz matrices have been shown to exist.

- In addition to lending a new insight into the continuous-time and discrete-time bilateral symmetric equations, our Sylvester algorithms differ from already published methods on the following key points: they are numerically reliable, rely upon well-worked numerical tools and can straightforwardly handle zeros on the unit circle.

Some possible extensions and future research directions are now mentioned.

- The Sylvester matrix approach results in a linear system of equations that may feature high dimensions. Recent works on resolution of Toeplitz systems of equations hint of a possible method for solving these systems in a numerically reliable way while simultaneously taking advantage of the problem structure. It may result in a reduced computational cost.
- Similarly, the block Vandermonde structure of the interpolated matrix should be exploited. Most notably, this would be aimed to overcoming the potential numerical instability naturally introduced by the ill-conditioning of the Vandermonde matrix.
- The matrix pencil approach also seems not to have been sufficiently exploited. Using for instance the theoretical developments recently exposed in [43], it should be highly instructive to transcribe typical polynomial matrix problems in terms of matrix pencil problems. Moreover, in addition to providing useful numerical tools for the control community, the conception of new algorithms also usually lends insight into fundamental problems
- The generalization of our approach to other fundamental polynomial matrix problems is also underway. There is still a dramatic need of numerically reliable algorithms for performing really basic tasks such as, for instance, the reduction of a polynomial matrix into Smith or diagonal form. Recent results obtained in the scope of symbolic computation may prove useful for this purpose [50].
- The development of a mathematical theory on the stability of polynomial operations is required. Concepts such as backward of forward error analysis [15, 16] should be extended to polynomials, thus allowing to rigorously prove the potential numerical instability of some algorithms on polynomials.
- Whereas general definitions of minimal polynomial basis where early provided in [13], a unifying theory of minimal degree solutions to linear matrix polynomial equations is still missing.
- A point that deserves further attention is the use of complex interpolation points in our algorithms, and more generally the possibility of working with complex polynomial matrix equations.

We hope that the achievements described in this thesis will open up new avenues for the development of efficient routines dealing with polynomial matrices.

Chapter 6

Nové poznatky (Original results)

The thesis features several results that are believed to be new. Our main contributions are the first numerically reliable methods ever published for

- evaluating the rank of an arbitrary polynomial matrix, see Section 4.3 and [P4]
- triangularizing an arbitrary polynomial matrix, see Section 4.4 and [P1, P5]
- solving the continuous-time symmetric bilateral equation, see Section 4.5 and [P3]
- solving the discrete-time symmetric bilateral equation, see Section 4.6 and [P6, P7].

Chapter 7

Konkrétní závěry pro praxi (Practical impact)

The new algorithms presented in this paper have been implemented and tested experimentally in the scope of the Polynomial Toolbox for MATLAB [P8, P10].

The Polynomial Toolbox is a package that may be used to perform computations with and on polynomial matrices. It allows

- to define polynomial matrices
- to display and plot polynomial matrices in many ways including 2D and 3D color plots
- to do operations on polynomial matrices such as transformation to various canonical forms including Hermite, Smith, echelon, row, column and diagonally reduced form
- to compute determinants and zeros of polynomial matrices
- to do operations among polynomial matrices, such as addition and multiplication, and finding least common multiples and greatest common divisors
- to perform advanced operations such as spectral factorization and computing null spaces
- to use polynomial matrices for modeling linear time-invariant systems, and to transform polynomial representations to and from other representations
- to solve many types of linear polynomial matrix equations
- to solve various types of control problems for linear systems, such as pole placement, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ optimization
- to solve various problems of signal processing

The Toolbox is aimed at industrial as well as classroom use in the fields of control systems, signal processing and applied mathematics. Its current version consists of almost one hundred files. Many of them are based on the algorithms presented in this thesis.

A pre-commercial version of the Polynomial Toolbox has been released on July 1, 1997. It is now available as freeware through the Toolbox WWW home page at the URL `http://www.math.utwente.nl/polbox`. Future users are welcome to register and download the Toolbox along with its hypertext manual and tutorial. They can also obtain more information about the Toolbox, its applications in various engineering fields and university courses, numerical algorithms and future plans.

Chapter 8

Ohlas autorových výsledků (Feedback)

The new algorithms developed are published in a series of scientific papers that are currently accepted for publication in leading international technical journals such as Automatic (IFAC) and IEEE Transactions on Automatic Control. Some of them have already been presented at prestigious international conferences (ECC 97). Immediate reaction of scientific community is highly positive but it is too early to register real citations of the papers.

Next to the standard publications, all the algorithms have been implemented in MATLAB code as a part of the Polynomial Toolbox for MATLAB, substantially improving its efficiency. The Polynomial Toolbox is a software package for industrial design of control systems and signal processing filters. Its pre-commercial version has attracted high interest of major industrial companies. The Toolbox home-page registers almost 3000 visits and more than 500 users from leading universities and industrial companies including Daimler-Benz, Kodak, Ford, Boeing, Motorola, Ericsson, ...

The American company Wolfram Research producing MATHEMATICA, a major competitor to MATLAB, has recently expressed an interest in having all the algorithms programmed also in the MATHEMATICA code.

The algorithms are also distributed within European Community through the European Network of Excellence EUROPOLY. This network is aimed at industrial applications of polynomial methods. It consists of 14 European companies and universities supported by EC project CP97:7010. The network is participated by several industrial companies and remains open to others that can join it later.

Seznam publikací (Publications)

[P1] D. Henrion and M. Šebek, Efficient Algorithms for Polynomial Matrix Triangularization, to appear in the *IEEE Transactions on Automatic Control*, 1999.

[P2] M. Šebek, D. Henrion and H. Kwakernaak, New Algorithms for Polynomial Matrices, *Proceedings of the IEEE Conference on Decision and Control*, Tampa, Florida, December 1998.

[P3] D. Henrion and M. Šebek, Symmetric Matrix Polynomial Equation : Interpolation Results, *Automatica*, Vol. 34, No. 7, pp. 811–824, 1998.

[P4] D. Henrion and M. Šebek, Numerical Methods for Polynomial Rank Evaluation, *Proceedings of the IFAC Conference on System Structure and Control*, Nantes, France, July 1998.

[P5] D. Henrion and M. Šebek, Efficient Algorithms for Polynomial Matrix Triangularization, *Proceedings of the IEEE Mediterranean Conference on Control and Automation*, Alghero, Sardinia, Italy, June 1998.

[P6] D. Henrion and M. Šebek, An Efficient Numerical Method for the Discrete-Time Symmetric Matrix Polynomial Equation, *IEE Proceedings, Control Theory and Applications*, Vol. 145, No. 5, pp. 443–448, 1998.

[P7] D. Henrion and M. Šebek, An Efficient Numerical Method for the Discrete-Time Symmetric Matrix Polynomial Equation, *Proceedings of the European Control Conference*, Brussels, Belgium, Vol. 6, Part B, FR-A H6, July 1997.

[P8] M. Šebek, S. Pejchová, R.C.W. Strijbos and D. Henrion, Polynomial Matrix Operations in MATLAB, *Proceedings of the International Conference on Computer Aided Systems Theory and Technology EUROCAST*, Las Palmas de Gran Canaria, Spain, February 1997.

[P9] M. Šebek, F. Kraus, S. Pejchová, H. Kwakernaak and D. Henrion, Numerical Methods for Zeros and Determinant of Polynomial Matrix, *Proceedings of the IEEE Mediterranean Symposium on New Directions in Control and Automation*, pp. 488-491, Chania, Crete, Greece, June 1996.

[P10] D. Henrion, F. Kraffer, H. Kwakernaak, S. Pejchová, M. Šebek and R. C. W. Strijbos, The Polynomial Toolbox for MATLAB, Version 1.5.

Literatura (References)

[1] B. D. O. Anderson, From Youla-Kučera to Identification, Adaptive and Nonlinear Control, *Proceedings of the IFAC World Congress*, San Francisco, California, pp. 39–59, 1996.

[2] P. J. Antsaklis and Z. Gao, Polynomial and Rational Matrix Interpolation: Theory and Control Applications, *International Journal of Control*, Vol. 58, No. 1, pp. 349–404, 1993.

[3] A. E. Barabanov, Canonical Matrix Factorization and Polynomial Riccati Equations, *European Journal of Control*, Vol. 3, No. 1, pp. 47–67, 1997.

[4] S. Barnett, Polynomial and Linear Control Synthesis, *Marcel Dekker*, London, 1983.

[5] T. G. J. Beelen and G. W. Veltkamp, Numerical Computation of a Coprime Factorization of a Transfer Function Matrix, *Systems and Control Letters*, Vol. 9, pp. 281–288, 1987.

[6] T. G. J. Beelen and P. Van Dooren, A Pencil Approach for Embedding a Polynomial Matrix into a Unimodular Matrix, *SIAM Journal on Matrix Analysis and Applications*, Vol. 9, No. 1, pp. 77–89, 1988.

[7] T. G. .J. Beelen, G. J. Van Den Hurk and C. Praagman, A New Method for Computing a Column Reduced Polynomial Matrix, *Systems & Control Letters*, Vol. 10, pp. 217–224, 1988.

[8] B. R. Bitmead, S. Y. Kung, B. D. O. Anderson and T. Kailath, Greatest Common Divisors via Generalized Sylvester and Bézout Matrices, *IEEE Transactions on Automatic Control*, Vol. 23, No. 6, pp. 1043–1047, 1978.

[9] S. Boyd, L. El Ghaoui, E. Féron and V. Balakrishnan, Linear Matrix Inequalities in System and Control Theory, *SIAM Studies in Applied Mathematics*, Vol. 15, Philadelphia, 1994.

[10] F. M. Callier and C. A. Desoer, Multivariable Feedback Systems, *Springer Verlag*, Berlin, 1982.

[11] C. Chen, Linear System Theory and Design, *Holt, Rinehart and Winston*, New York, 1984.

[12] E. Emre, The Polynomial Equation $QQ_c + RP_c = \Phi$ with Application to Dynamic Feedback, *SIAM Journal on Control and Optimization*, Vol. 18, pp. 611–620, 1984.

[13] G. D. Forney Jr., Minimal Bases of Rational Vector Spaces with Applications to Multivariable Linear Systems, *SIAM Journal on Control*, Vol. 13, No. 3, pp. 493–520, 1975.

[14] I. Gil, Contribution à l'Algèbre Linéaire Formelle, Formes Normales de Matrices et Applications (in French), *PhD Thesis*, Institut National Polytechnique de Grenoble, Grenoble, France, 1993.

[15] G. H. Golub and C. F. Van Loan, Matrix Computations, Second Edition, *The Johns Hopkins University Press*, Baltimore, 1989.

[16] N. J. Higham, Accuracy and Stability of Numerical Algorithms, *SIAM*, Philadelphia, 1996.

[17] K. J. Hunt and M. Šebek, Optimal Multivariable Regulation with Disturbance Measurement Feedforward, *International Journal of Control*, Vol. 49, No. 1, pp. 373–378, 1989.

[18] K. J. Hunt and V. Kučera, The Standard $\mathcal{H}_2$-Optimal Control Problem: A Polynomial Solution, *International Journal of Control*, Vol. 56, No. 1, pp. 245–251, 1992.

[19] J. Ježek and V. Kučera, Efficient Algorithm for Matrix Spectral Factorization, *Automatica*, Vol. 21, No. 6, pp. 663–669, 1985.

[20] J. Ježek, Symmetric Matrix Polynomial Equations. *Kybernetika*, Vol. 22, No. 1, pp. 19–30, 1986.

[21] J. Ježek and K. J. Hunt, Coupled Polynomial Equations for LQ Control Synthesis and an Algorithm for Solution, *International Journal of Control*, Vol. 58, No. 5, pp. 1155–1167, 1993.

[22] T. Kailath, Linear Systems, *Prentice Hall*, Englewood Cliffs, 1980.

[23] E. Kaltofen, M. S. Krishnamoorthy and B. D. Saunders, Fast Parallel Computations of Hermite and Smith Forms of Polynomial Matrices, *SIAM Journal of Algebraic and Discrete Methods*, Vol. 8, pp. 683–690, 1987.

[24] N. Karcanias and M. Mitrouli, A Matrix Pencil Based Numerical Method for the Computation of the GCD of Polynomials, *IEEE Transactions on Automatic Control*, Vol. 39, No. 5, pp. 977–981, 1994.

[25] F. Kraffer, S. Pejchová and M. Šebek, Polynomial Operations: Numerical Performance in Matrix Diophantine Equations, *Kybernetika*, Vol. 30, No. 6, pp. 745–753, 1994.

[26] F. Kraus and V. Kučera, Robust Assignment of Polynomial Matrix Polytopes, *Proceedings of the European Control Conference*, Brussels, Belgium, July 1997.

[27] V. Kučera, Discrete Linear Control, The Polynomial Equation Approach, *Wiley*, Chichester, 1979.

[28] V. Kučera and M. Šebek, On Deadbeat Controllers, *IEEE Transactions on Automatic Control*, Vol. 29, No. 8, pp. 719–722, 1984.

[29] V. Kučera, Stationary LQG Control of Singular Systems, *IEEE Transactions on Automatic Control*, Vol. 31, No. 1, pp. 31–39, 1986.

[30] V. Kučera, J. Ježek and M. Krupička, Numerical Analysis of Diophantine Equations, *Proceedings of the UK-Czech Seminar on Adaptive Control*, Prague, Czech Republic, 1990.

[31] V. Kučera, Analysis and Design of Discrete Linear Control Systems, *Prentice Hall*, London, 1991.

[32] V. Kučera and P. Zagalak, Constant Solutions of Polynomial Equations, *Internation Journal of Control*, Vol. 53, No. 2, pp. 495-502, 1991.

[33] V. Kučera, Fixed Degree Solutions of Polynomial Equations, *IFAC Workshop on Systems, Structure and Control*, Prague, Czechoslovakia, pp. 24–26, 1992.

[34] V. Kučera, Diophantine Equations in Control - a Survey, *Automatica*, Vol. 29, No. 6, pp. 1361–1375, 1993.

[35] H. Kwakernaak, The Polynomial Approach to $\mathcal{H}_\infty$-optimal regulation, In E. Mosca and L. Pandolfi (Editors), $\mathcal{H}_\infty$-Control Theory, Como, 1990, *Lecture Notes in Mathematics*, Vol. 1496, Springer Verlag, 1990.

[36] H. Kwakernaak, Robust Control and $\mathcal{H}_\infty$-optimization – Tutorial Paper, *Automatica*, Vol. 29, No. 2, pp. 255–273, 1993.

[37] H. Kwakernaak and M.Šebek, Polynomial J-Spectral Factorization, *IEEE Transactions on Automatic Control*, Vol. 39, pp. 315–328, 1994.

[38] H. Kwakernaak, State-Space Algorithms for Polynomial Matrix Computations, *EURACO Workshop on Recent Results in Robust and Adaptive Control*, Florence, Italy, September 1995.

[39] S. Labhalla, H. Lombardi and R. Marlin, Algorithmes de Calcul de la Réduction de Hermite d'une Matrice à Coefficients Polynomiaux (in French), *Theoretical Computer Science*, Vol. 161, pp. 69–92, 1996.

[40] P. Lancaster and M. Tismenetsky, The Theory of Matrices, Second Edition with Applications, *Academic Press*, New York, 1985.

[41] A. J. Laub, A Schur Method for Solving Algebraic Riccati Equations, *IEEE Transactions on Automatic Control*, Vol. 24, No. 6, pp. 913–921, 1979.

[42] A. B. Özgüler, Linear Multichannel Control - A System Matrix Approach, *Prentice Hall*, New York, 1994.

[43] M. P. Quéré-Stuchlik, Algorithmique des Faisceaux Linéaires de Matrices, Application à la Théorie des Systèmes Linéaires et à la Résolution d'Équations Algébro-Différentielles (in French), *PhD Thesis*, Université Paris VI, Paris, France, 1997.

[44] M. Šebek, K. J. Hunt and M. J. Grimble, LQG Regulation with Disturbance Measurement Feedforward, *International Journal of Control*, Vol. 47, No. 5, pp. 1497–1505, 1988.

[45] T. Söderström, J. Ježek and V. Kučera, An Efficient and Versatile Algorithm for Computing the Covariance Function of an ARMA Process, *to appear in Automatica*, 1998.

[46] P. Stefanidis, A. P. Papliński and M. J. Gibbard, Numerical Operations with Polynomial Matrices: Application to Multi-Variable Dynamic Compensator Design, *Lecture Notes in Control and Information Sciences*, Vol. 171, Springer Verlag, Berlin, 1992.

[47] A. Storjohann, Computation of Hermite and Smith Normal Forms of Matrices, *Master's Thesis*, University of Waterloo, Ontario, Canada, 1994.

[48] P. Van Dooren, The Computation of Kronecker's Canonical Form of Singular Pencil, *Linear Algebra and Applications*, Vol. 27, pp. 103–141, 1979.

[49] M. Vidyasagar, Control Systems Synthesis: a Factorization Approach, *MIT Press*, Boston, 1985.

[50] G. Villard, Generalized Subresultants for Computing the Smith Normal Form of Polynomial Matrices, *Journal of Symbolic Computation*, Vol. 20, pp. 269–286, 1995.

[51] G. Villard, Computing Popov and Hermite Forms of Polynomial Matrices, *International Symposium on Symbolic and Algebraic Computation*, Zurich, Switzerland, ACM Press, pp. 250–258, 1996.

[52] J. C. Willems, From Time Series to Linear System, Part I - Finite Dimensional Linear Time-Invariant Systems, *Automatica*, Vol. 22, pp. 561–580, 1986.

[53] W. A. Wolovich, Skew Prime Polynomial Matrices, *IEEE Transactions on Automatic Control*, Vol. 23, No. 5, pp. 880–887, 1978.

[54] W. A. Wolovich and P. J. Antsaklis, The Canonical Diophantine Equations with Applications, *SIAM Journal on Control and Optimization*, Vol. 22, No. 5, pp. 777–787, 1984.

[55] W. M. Wonham, Linear Multivariable Control - a Geometric Approach, *Springer Verlag*, Berlin, 1985.