An Efficient Numerical Method for the Discrete-time Symmetric Matrix Polynomial Equation *†

Didier Henrion[1‡] Michael Šebek[2,3]

[1] L.A.A.S.-C.N.R.S.
7 Avenue du Colonel Roche
31 077 Toulouse, Cedex 4, France

[2] Trnka Laboratory of Automatic Control
Faculty of Electrical Engineering
Czech University of Technology
Prague, Czech Republic

[3] Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Prague, Czech Republic

Abstract

A numerical procedure is proposed to solve a matrix polynomial equation frequently encountered in discrete-time control and signal processing. The algorithm is based on a simple rewriting of the original equation in terms of a reduced Sylvester matrix. In contrast to previously published methods, it does not make use of elementary polynomial operations. Moreover and most notably, it is numerically reliable. Basic examples borrowed from control and signal processing literature are aimed at illustrating the simplicity and efficiency of this new numerical method.

1 Introduction

We consider the discrete-time symmetric matrix polynomial equation

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

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

$$\begin{array}{rcl} \mathbf{B}(\mathtt{d}) & = & \mathbf{B}_l(\mathtt{d}^{-1}) + \mathbf{B}_r(\mathtt{d}) \\ \mathbf{B}_l(\mathtt{d}) & = & \mathbf{B}_r'(\mathtt{d}) \end{array} \tag{2}$$

and $\mathbf{X}(\mathtt{d})$ 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.

*A short version of this paper was presented at the European Control Conference, Brussels, Belgium, July 1997.

†The main part of this work has been carried out while the first author was with the Institute of Information Theory and Automation, Prague, Czech Republic. This work is part of the EC Copernicus Project entitled *Numerical algorithms for CAE based on modern polynomial control methods* and was supported by the European Community Grant No. CP/93:2424. It is also part of the Barrande Project No. 97/005-97/026 and was supported by the Grant Agency of the Czech Republic under contract No. 102/97/0861, by the Ministry of Eduction of the Czech Republic under contract No. VS97/034 and by the French Ministry of Education and Research under contract No. 10-INSA-96.

‡Corresponding author. Fax: (+33 5) 61 33 69 69. E-mail: `henrion@laas.fr`.

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 [7]. In [4], a spectral factorization algorithm based on a Newton-Raphson scheme consisting of equation (1) at every iteration was proposed.

Another significant role played by this equation stems from signal processing. In [10] it is shown that equation (1) 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 (1), only a few works have been devoted so far to its resolution by means of systematic numerical procedures. In [4], the authors used an algorithm based on polynomial reductions that has been studied in further detail in [5]. 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[1], a more efficient alternative is required. A different approach has been followed in [10] where it is shown on a simple example that solving (1) 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 (1).

Based on the above considerations, the main contribution of our paper is a numerically reliable algorithm that solves (1). 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 [4, 5].

The paper is organized as follows. In Section 2, an equivalent constant matrix formulation is obtained by equating coefficients of powers of $\mathtt{d}$ in (1). This technique is usually referred to as the Sylvester matrix approach to solving polynomial equations. Moreover, symmetry properties (2) 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. In Section 3, we show that under some assumptions on $\mathbf{A}(\mathtt{d})$, existence as well as uniqueness of solution to (1) can be guaranteed. In Section 4, three simple numerical examples borrowed from control and signal processing literature underline both simplicity and efficiency of this new numerical method. Finally, in Section 5, numerical advantages of our algorithm over previously published methods are illustrated by intensive computational experiments.

2 Reduced Sylvester Matrix Formulation

In this section, we aim to show that matrix polynomial equation (1) can be solved without performing any polynomial operations. For that purpose, we will establish a bijective correspondence between every solution to (1) and a constant vector solution to a linear system of equations which order can be kept reasonable thanks to symmetry properties (2). But first, we require the Sylvester matrix of a given polynomial matrix to be defined.

2.1 Sylvester Matrix

Any polynomial matrix $\mathbf{F}(\mathtt{d}) = \mathbf{F}_0 + \mathbf{F}_1\mathtt{d} + \cdots + \mathbf{F}_{d_F}\mathtt{d}^{d_F}$ can be written

$$\begin{aligned} \mathbf{F}(\mathtt{d}) &= ([1\ \mathtt{d} \ldots \mathtt{d}^{d_F}] \otimes \mathbf{I}) \begin{bmatrix} \mathbf{F}_0 \\ \mathbf{F}_1 \\ \vdots \\ \mathbf{F}_{d_F} \end{bmatrix} \\ &= (\mathbf{D}_F(\mathtt{d}) \otimes \mathbf{I})\mathbf{F}. \end{aligned}$$

[1] MATLAB is a trademark of The MathWorks, Inc.

Now if $\mathbf{G}(\mathtt{d})$ and $\mathbf{H}(\mathtt{d})$ are two polynomial matrices of respective degrees d_G and d_H such that

$$\mathbf{F}(\mathtt{d}) = \mathbf{G}(\mathtt{d})\mathbf{H}(\mathtt{d})$$

and $d_G + d_H = d_F$, then

$$\mathbf{F} = \underbrace{\begin{bmatrix} \mathbf{G}_0 & & & \mathbf{0} \\ \mathbf{G}_1 & \mathbf{G}_0 & & \\ \vdots & \mathbf{G}_1 & \ddots & \\ \mathbf{G}_{d_G} & \vdots & & \mathbf{G}_0 \\ & \mathbf{G}_{d_G} & & \mathbf{G}_1 \\ & & \ddots & \vdots \\ \mathbf{0} & & & \mathbf{G}_{d_G} \end{bmatrix}}_{<\mathbf{G}(\mathtt{d})>_{d_H}} \underbrace{\begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_{d_H} \end{bmatrix}}_{\mathbf{H}}$$

where the $(d_G + d_H + 1)$-by-$(d_H + 1)$ block-matrix $< \mathbf{G}(\mathtt{d}) >_{d_H}$ is called *Sylvester matrix* of order d_H of $\mathbf{G}$. That is, the column block-matrix form of the product of two polynomial matrices can be obtained via multiplication of two constant block-matrices. This method can readily be generalized to two-sided polynomial matrices since if

$$\mathbf{R}(\mathtt{d}) = \mathbf{R}_{-\alpha}\mathtt{d}^{-\alpha} + \cdots + \mathbf{R}_{-1}\mathtt{d}^{-1} + \mathbf{R}_0 + \mathbf{R}_1\mathtt{d} + \cdots + \mathbf{R}_\beta\mathtt{d}^\beta$$

for some integers $\alpha, \beta > 0$, we can work with polynomial matrix

$$\tilde{\mathbf{R}}(\mathtt{d}) = \mathtt{d}^\alpha\mathbf{R}(\mathtt{d}) = \mathbf{R}_{-\alpha} + \mathbf{R}_{-\alpha+1}\mathtt{d} + \cdots + \mathbf{R}_\beta\mathtt{d}^{\alpha+\beta}.$$

In that case, some care must be taken to handle the degree offset α introduced in $\tilde{\mathbf{R}}(\mathtt{d})$, in particular when performing rational matrix additions or multiplications.

Note that the above results are well-known and can be found in any standard reference on control theory [6].

2.2 Equivalent Linear System of Equations

First, equation (1) can be written in terms of Kronecker product and vectors (see for instance Chapter 12 in [9])

$$[\mathbf{I}_n \otimes \mathbf{A}'(\mathtt{d}^{-1})]\overrightarrow{\mathbf{X}}(\mathtt{d}) + [\mathbf{A}'(\mathtt{d}) \otimes \mathbf{I}_n]\overrightarrow{\mathbf{X}'}(\mathtt{d}^{-1}) = \overrightarrow{\mathbf{B}}(\mathtt{d}) \tag{3}$$

where $\overrightarrow{\mathbf{X}}(\mathtt{d}) = \overrightarrow{\mathbf{X}_0} + \overrightarrow{\mathbf{X}_1}\mathtt{d} + \cdots + \overrightarrow{\mathbf{X}_{d_X}}\mathtt{d}^{d_X}$ is a polynomial vector obtained by stacking together columns of polynomial matrix $\mathbf{X}(\mathtt{d})$.

Then introducing $\mathbf{P}$ as the square permutation matrix such that

$$\overrightarrow{\mathbf{X}'}(\mathtt{d}) = \mathbf{P}\overrightarrow{\mathbf{X}}(\mathtt{d}), \tag{4}$$

matrices

$$\begin{aligned} \mathbf{P}_+ &= \mathbf{I}_n + \mathbf{P} \\ \mathbf{P}_- &= \mathbf{I}_n - \mathbf{P} \end{aligned}$$

are called *projectors* [9]. One can check that $\mathrm{rank}\mathbf{P}_+ = n(n+1)/2$ and $\mathrm{rank}\mathbf{P}_- = n(n-1)/2$. Thus if $\tilde{\mathbf{P}}_+$ is a matrix whose $n(n+1)/2$ rows generate a basis for $\mathrm{Im}\mathbf{P}_+$ and if $\tilde{\mathbf{P}}_-$ is a matrix whose $n(n-1)/2$ rows generate a basis for $\mathrm{Im}\mathbf{P}_-$, it can easily be deduced that

$$\mathbf{\Pi} = \begin{bmatrix} \tilde{\mathbf{P}}_+ \\ \tilde{\mathbf{P}}_- \end{bmatrix} \tag{5}$$

is a non-singular matrix and moreover that $\frac{1}{2}\tilde{\mathbf{P}}_+\mathbf{P}_+ = \tilde{\mathbf{P}}_+$ and $\frac{1}{2}\tilde{\mathbf{P}}_-\mathbf{P}_- = \tilde{\mathbf{P}}_-$.

According to commutativity properties of Kronecker product [9], equation (3) yields

$$\mathbf{Q}(\mathtt{d}^{-1})\overrightarrow{\mathbf{X}}(\mathtt{d}) + \mathbf{P}\mathbf{Q}(\mathtt{d})\overrightarrow{\mathbf{X}}(\mathtt{d}^{-1}) = \overrightarrow{\mathbf{B}_r}(\mathtt{d}) + \mathbf{P}\overrightarrow{\mathbf{B}_r}(\mathtt{d}^{-1}) \tag{6}$$

for

$$\mathbf{Q}(\mathtt{d}) = \mathbf{I}_n \otimes \mathbf{A}'(\mathtt{d}) = \mathbf{Q}_0 + \mathbf{Q}_1\mathtt{d} + \cdots \mathbf{Q}_{d_A}\mathtt{d}^{d_A}.$$

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

With these conventions and the results recalled in the previous section, the Sylvester matrix formulation of matrix polynomial equation (6) reads

$$\mathcal{A}\underbrace{\begin{bmatrix} \overrightarrow{\mathbf{X}_0} \\ \overrightarrow{\mathbf{X}_1} \\ \vdots \\ \overrightarrow{\mathbf{X}_{d_X}} \end{bmatrix}}_{\overrightarrow{\mathbf{X}}} + \tilde{\mathcal{A}}\begin{bmatrix} \overrightarrow{\mathbf{X}_{d_X}} \\ \vdots \\ \overrightarrow{\mathbf{X}_1} \\ \overrightarrow{\mathbf{X}_0} \end{bmatrix} = \begin{bmatrix} \mathbf{P}\overrightarrow{\mathbf{B}_{r\,d_B}} \\ \vdots \\ \mathbf{P}\overrightarrow{\mathbf{B}_{r\,1}} \\ \mathbf{P}_+\overrightarrow{\mathbf{B}_{r\,0}} \\ \overrightarrow{\mathbf{B}_{r\,1}} \\ \vdots \\ \overrightarrow{\mathbf{B}_{r\,d_B}} \end{bmatrix} = \begin{bmatrix} \mathcal{B} \\ \mathbf{P}_+\overrightarrow{\mathbf{B}_{r\,0}} \\ \mathcal{B}^* \end{bmatrix} \tag{7}$$

where

$$\mathcal{A} = \begin{bmatrix} < \mathtt{d}^{d_A}\mathbf{Q}(\mathtt{d}^{-1}) >_{d_X} \\ \mathbf{0}_{n^2(2d_B - d_A - d_X)\times n^2(d_X+1)} \end{bmatrix} \tag{8}$$

and

$$\tilde{\mathcal{A}} = \begin{bmatrix} \mathbf{0}_{n^2(2d_B - d_A - d_X)\times n^2(d_X+1)} \\ < \mathbf{P}Q(\mathtt{d}) >_{d_X} \end{bmatrix}. \tag{9}$$

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 (7) such that unknown scalar coefficients of $\mathbf{X}(\mathtt{d})$ appear only in the vector $\overrightarrow{\mathbf{X}}$:

$$\mathbf{T}\begin{bmatrix} \overrightarrow{\mathbf{X}} \\ \overrightarrow{\mathbf{X}} \end{bmatrix} = \begin{bmatrix} \mathcal{B} \\ \mathbf{P}_+\overrightarrow{\mathbf{B}_{r\,0}} \\ \mathcal{B}^* \end{bmatrix} \tag{10}$$

for

$$\mathbf{T} = [\mathcal{A} \quad \tilde{\mathcal{A}}(\mathbf{J}_{d_X} \otimes \mathbf{I}_{n^2})]. \tag{11}$$

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

$$\frac{1}{2}\begin{bmatrix} \mathbf{I}_{n^2 d_B} & \mathbf{0} & \mathbf{J}_{d_B} \otimes \mathbf{P} \\ \mathbf{0} & \mathbf{\Pi} & \mathbf{0} \\ -\mathbf{I}_{n^2 d_B} & \mathbf{0} & \mathbf{J}_{d_B} \otimes \mathbf{P} \end{bmatrix} \mathbf{T} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathcal{Q} & \mathbf{0} \\ \tilde{\mathbf{P}}_+\mathcal{Q} & \\ \mathbf{0} & \tilde{\mathbf{P}}_-\mathcal{Q} \\ & -\mathcal{Q} \end{bmatrix} = \begin{bmatrix} \mathcal{Q}_R & \mathbf{0} \\ \mathbf{0} & \mathcal{Q}_R^* \end{bmatrix} \tag{12}$$

where $\mathcal{Q} = [\mathbf{Q}_0\ \mathbf{Q}_1\ \cdots\ \mathbf{Q}_{d_X}]$ and $\mathbf{Q}_k = \mathbf{0}$ if $k > d_A$.

Since $\mathcal{B} = (\mathbf{J}_{d_B} \otimes \mathbf{P})\mathcal{B}^*$, substitution of (12) into (10) gives

$$\begin{bmatrix} \mathcal{Q}_R & \mathbf{0} \\ \mathbf{0} & \mathcal{Q}_R^* \end{bmatrix}\begin{bmatrix} \overrightarrow{\mathbf{X}} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathcal{B} \\ \tilde{\mathbf{P}}_+\overrightarrow{\mathbf{B}_{r\,0}} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathcal{B}_R \\ \mathbf{0} \end{bmatrix} \tag{13}$$

which in turn yields

$$\mathcal{Q}_R\overrightarrow{\mathbf{X}} = \mathcal{B}_R \tag{14}$$

by considering the non-zero rows in (13). Note that $\mathcal{Q}_R$ has dimensions $n^2 d_B + n(n+1)/2$ and $n^2(d_X+1)$. Because (12) is an equivalence relation involving non-singular matrices, there exists a one-to-one correspondence between every solution $\mathbf{X}(\mathtt{d})$ to matrix polynomial equation (1) and the constant vector solution $\overrightarrow{\mathbf{X}}$ of linear system of equations (14). The following theorem has been shown.

Theorem 2.1 *Equation (1) has a polynomial solution* $\mathbf{X}(\mathtt{d})$ *if and only if there exists a constant vector solution* $\overrightarrow{\mathbf{X}}$ *to the reduced linear system of equation (14).*

2.3 Algorithm

Input	Polynomial matrices $\mathbf{A}(\mathtt{d})$, $\mathbf{B}(\mathtt{d})$ verifying symmetry properties (2).
Output	Polynomial matrix $\mathbf{X}(\mathtt{d})$ solution to matrix polynomial equation (1).

Step 1	Build matrix $\mathbf{T}$ in (11) from relations (8) and (9).
Step 2	Using equations (5) and (12), extract the reduced Sylvester matrix $\mathcal{Q}_R$.
Step 3	Build matrix $\mathcal{B}_R$ according to equations (7) and (13).
Step 4	Every constant vector solution $\overrightarrow{\mathbf{X}}$ to linear system (14) corresponds to a polynomial matrix solution $\mathbf{X}(\mathtt{d})$ to matrix polynomial equation (1).

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 [2]. Furthermore, symmetry properties (2) have been considered so as to significantly reduce the original problem dimensions.

3 Existence and Shape of Solution

Since we dispose now of an efficient method to solve equation (1), it is natural to wonder whether it has a solution or not. In this optic, Ježek [5] has shown the following useful result.

Theorem 3.1 *Suppose that matrix* $\mathbf{A}(\mathtt{d})$ *is stable in the discrete-time sense, i.e.* $\det \mathbf{A}(\mathtt{d}) \neq 0$ *for* $|\mathtt{d}| \leq 1$. *Then matrix polynomial equation (1) has a solution. Furthermore, if* $\mathbf{A}(0)$ *is upper triangular and non-singular, there exists a* unique *solution* $\mathbf{X}(\mathtt{d})$ *to (1) with* $\mathbf{X}(0)$ *upper triangular.*

The proof of Theorem 3.1 provided in [5] is based on elementary polynomial operations, but the results can be translated in terms of properties verified by linear system (14) in the following way.

Corollary 3.1 *If* $\mathbf{A}(\mathtt{d})$ *is stable in the discrete-time sense, then*

$$\mathrm{rank}[\mathcal{Q}_R \quad \mathcal{B}_R] = \mathrm{rank}\mathcal{Q}_R. \tag{15}$$

Moreover if $\mathbf{A}(0)$ *is upper triangular, then after insertion of* $n(n-1)/2$ *supplementary linear equations setting the strictly lower part of* $\mathbf{X}(0)$ *to zero, linear system (14) reads*

$$\begin{bmatrix} \mathcal{Q}_R \\ \mathbf{E} \end{bmatrix} \overrightarrow{\mathbf{X}} = \mathcal{Q}_T \overrightarrow{\mathbf{X}} = \begin{bmatrix} \mathcal{B}_R \\ \mathbf{0} \end{bmatrix} = \mathcal{B}_T \tag{16}$$

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

Proof : Assuming $\deg \mathbf{X}(\mathtt{d}) = \deg \mathbf{B}_r(\mathtt{d}) = d_B$, rank condition (15) and non-singularity of matrix $\mathcal{Q}_T$ in (16) are immediate consequences of existence and uniqueness conditions of Theorem 3.1. □

The numerical aspect of Corollary 3.1 is especially interesting since the existence of a solution to linear system (14) is guaranteed *a priori*. That is, no preliminary rank tests have to be performed on matrices $\mathcal{Q}_R$ and $\mathcal{B}_R$.

4 Applications

4.1 Spectral Factorization

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

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

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

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

According to [4], the Newton-Raphson iterative process

- $\mathbf{A}'_k(\mathtt{d}^{-1})\mathbf{X}_k(\mathtt{d}) + \mathbf{X}'_k(\mathtt{d}^{-1})\mathbf{A}_k(\mathtt{d}) = \mathbf{B}(\mathtt{d})$
  for upper triangular $\mathbf{X}_k(0)$ (17)
- $\mathbf{A}_{k+1}(\mathtt{d}) = \frac{1}{2}(\mathbf{A}_k(\mathtt{d}) + \mathbf{X}_k(\mathtt{d}))$ (18)

for $k = 0, 1, 2, \ldots$ and a non-singular stable initial matrix $\mathbf{A}_0(\mathtt{d})$ such that $\mathbf{A}_0(0)$ is upper triangular, converges quadratically to a stable matrix $\mathbf{A}(\mathtt{d}) = \mathbf{A}_\infty(\mathtt{d})$ solution to the spectral factorization problem. Following [4], we choose $\mathbf{A}_0(\mathtt{d}) = \mathbf{\Psi}$ as the Cholesky square root matrix of the diagonal leading coefficient of $\mathbf{B}(\mathtt{d})$, viz. $\mathbf{\Psi}'\mathbf{\Psi} = \mathbf{B}(0)$. We will now show how equation (17) can be solved at a given step k, using linear system (16) and Corollary 3.1.

After $k = 2$ iterations of process (17-18), we obtain

$$\mathbf{A}_2(\mathtt{d}) = \begin{bmatrix} 4.0189 - 0.9925\mathtt{d} & -2.9528 \\ -0.9658\mathtt{d} & 5.0506 - 0.9875\mathtt{d} \end{bmatrix}$$

and we seek a solution $\mathbf{X}_2(\mathtt{d})$ to equation (17). Equivalent linear system (16) is easily obtained from equations (11) and (12). We have

$$\underbrace{\left[\begin{array}{cccccccc} -0.99 & -0.97 & 0 & 0 & 4.02 & 0 & 0 & 0 \\ 0 & -0.99 & 0 & 0 & 0 & 0 & 4.02 & 0 \\ 0 & 0 & -0.99 & -0.97 & -2.95 & 5.05 & 0 & 0 \\ 0 & 0 & 0 & -0.99 & 0 & 0 & -2.95 & 5.05 \\ 16.08 & 0 & 0 & 0 & -3.97 & -3.86 & 0 & 0 \\ -5.91 & 10.10 & 8.04 & 0 & 0 & -1.98 & -1.99 & -1.93 \\ 0 & 0 & -11.81 & 20.20 & 0 & 0 & 0 & -3.95 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]}_{\mathcal{Q}_T} \overrightarrow{\mathbf{X}_2} = \underbrace{\left[\begin{array}{c} -8 \\ 0 \\ -4 \\ -10 \\ 72 \\ -44 \\ 140 \\ \hline 0 \end{array}\right]}_{\mathcal{B}_T}.$$

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 3.1 assumptions hold and therefore $\mathcal{Q}_T$ is non-singular. The unique solution $\overrightarrow{\mathbf{X}_2}$ to linear system (16) yields

$$\mathbf{X}_2(\mathtt{d}) = \begin{bmatrix} 3.9820 - 1.0072\mathtt{d} & -3.0453 \\ -1.0324\mathtt{d} & 4.9516 - 1.0118\mathtt{d} \end{bmatrix}$$

that allows to build $\mathbf{A}_3(\mathtt{d})$, see equation (18).

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

$$\mathbf{A}_5(\mathtt{d}) = \begin{bmatrix} 4 - \mathtt{d} & -3 \\ -\mathtt{d} & 5 - \mathtt{d} \end{bmatrix}$$

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

4.2 Spectral Factorization: Critical Case

Even when $\mathbf{A}(\mathtt{d})$ does not verify the assumptions of Corollary 3.1, linear system (16) still provides a way to solve equation (1). However, preliminary rank tests are now mandatory to conclude on the existence of a solution.

To see that, let be given

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

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

$$\mathbf{B}_1(\mathtt{d}) = \begin{bmatrix} 2 & 2+2\mathtt{d} \\ 2\mathtt{d}^{-1}+2 & \mathtt{d}^{-2}+2\mathtt{d}^{-1}+2\mathtt{d}+\mathtt{d}^2 \end{bmatrix}$$

as the right hand side matrix in equation (1). Equivalent linear system (14) reads

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

Using basic numerical linear algebra tools, we solve the above system and find the following parameterization of all polynomial solutions $\mathbf{X}(\mathtt{d})$ to equation (1) :

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

for $a(\mathtt{d})$,$b(\mathtt{d})$ and $c(\mathtt{d})$ arbitrary scalar polynomials. In the same way, linear system (16) gives a parameterization of all polynomial solutions $\mathbf{X}(\mathtt{d})$ having an upper triangular absolute coefficient matrix $\mathbf{X}_0$. To see that, just set $c(\mathtt{d}) = 0$ in the previous expression.

If we choose at present

$$\mathbf{B}_2(\mathtt{d}) = \begin{bmatrix} 1 & 2+2\mathtt{d} \\ 2\mathtt{d}^{-1}+2 & \mathtt{d}^{-2}+2\mathtt{d}^{-1}+2\mathtt{d}+\mathtt{d}^2 \end{bmatrix}$$

as the right hand side matrix in equation (1), inspection of corresponding linear system (14) shows that $\mathrm{rank}\mathcal{Q}_R = 9$ while $\mathrm{rank}[\mathcal{Q}_R \quad \mathcal{B}_R] = 10$ and therefore that (1) does not admit any polynomial solution in this case.

4.3 Computing the Covariance Function of an ARMA Process

Consider the real ARMA process

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

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

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

Polynomial matrix $\mathbf{A}(\mathtt{d})$ 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

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

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

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

The computation follows [10] by partial fraction expansion of the spectral density in the form

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

This is equivalent to solving the symmetric matrix polynomial equation

$$\mathbf{X}(\mathtt{d})\mathbf{A}'(\mathtt{d}^{-1}) + \mathbf{A}(\mathtt{d})\mathbf{X}'(\mathtt{d}^{-1}) = \mathbf{C}(\mathtt{d})\mathbf{C}'(\mathtt{d}^{-1}) \tag{19}$$

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

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

by long polynomial division. Inspection shows that

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

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

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

Matrix polynomial equation (19) is actually a transposed version of (1), so that after performing the necessary transpositions our algorithm yields the linear system of equations

$$\begin{bmatrix} -2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -2.5 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 0 & 6 & 1 & 0 & 0 \\ 0 & 0 & 0 & -2.5 & 0 & 0 & 6 & 1 \\ 2 & 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 6 & 1 & 1 & 0 & 0 & -2.5 & -2 & 0 \\ 0 & 0 & 12 & 2 & 0 & 0 & 0 & -5 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \overrightarrow{\mathbf{X}_T} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Solution vector $\overrightarrow{\mathbf{X}_T} = [-0.167 \ \ 0 \ \ -1.5 \ \ -1.81 \ \ -0.333 \ \ -1 \ \ 0 \ \ -4.52]'$ provides a polynomial matrix solution to (19), namely

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

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

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

where $[\mathbf{A}(\mathtt{d})]^{-1}\mathbf{Y}(\mathtt{d})$ is strictly proper. We have used the matrix polynomial division macro of the Polynomial Toolbox [8] to get the covariance matrix function represented in Figure 1.

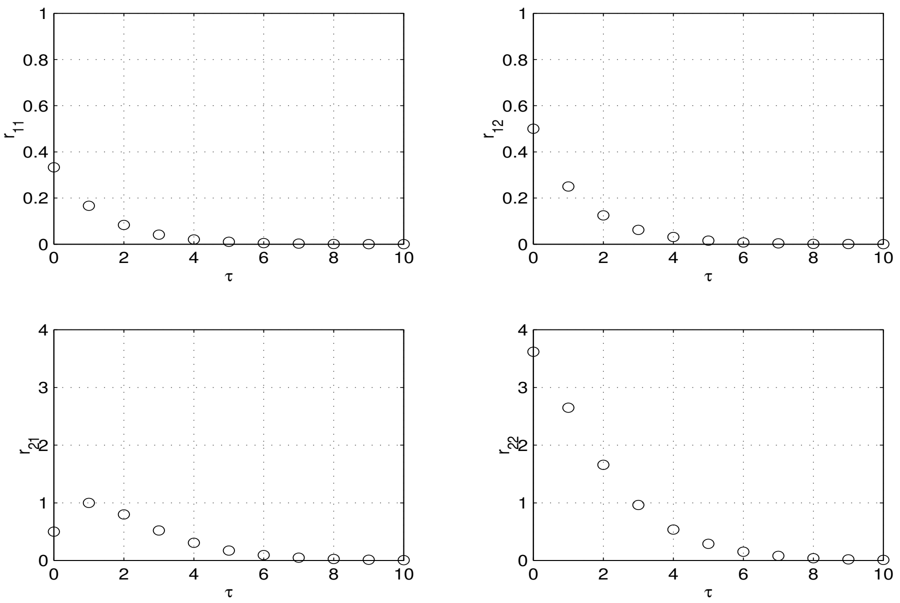

Figure 1: ARMA Process Covariance Matrix Function

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 in this paper) is based on Sylvester matrices. We simply invoked the MATLAB built-in Gaussian elimination algorithm to solve linear system (14).
- The polynomial reduction algorithm (presented in [4]) 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 [4]. 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 (1), that is equation (14) in [4].

For our computer simulations, $\mathbf{A}(\mathtt{d})$ and $\mathbf{X}(\mathtt{d})$ 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. $d_A = \deg \mathbf{A}(\mathtt{d}) = \deg \mathbf{X}(\mathtt{d}) = d_X$. Each constant scalar coefficient of polynomial entries of $\mathbf{A}(\mathtt{d})$ and $\mathbf{X}(\mathtt{d})$ was a Gaussian random variable with mean zero and variance two, except for the strictly lower triangular entries of $\mathbf{X}(\mathtt{d})$ that were set to zero. Matrix $\mathbf{B}(\mathtt{d})$ is evaluated from $\mathbf{A}(\mathtt{d})$ and $\mathbf{X}(\mathtt{d})$ using equation (1).

Input matrices $\mathbf{A}(\mathtt{d})$ and $\mathbf{B}(\mathtt{d})$ being given, each equation solver returns a solution $\hat{\mathbf{X}}_k(\mathtt{d})$ that can be compared with the known exact solution $\mathbf{X}(\mathtt{d})$ to equation (1). 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.

6 Conclusion

In this paper we proposed a Sylvester matrix approach to the discrete-time symmetric matrix polynomial equation and we described a new numerical procedure for its resolution.

In addition to lending a new insight into this type of bilateral polynomial equations, our algorithm differs from already published methods on the following key points :

- it is numerically reliable and faster
- it relies upon well-worked and readily available numerical tools
- zeros on the unit circle are handled without taking special precautions.

In a paper to be published [3], we propose a continuous-time version of this algorithm, based on Sylvester matrices and also on interpolation. These algorithms belong to a family of new numerical routines that are currently developed in the scope of the Polynomial Toolbox for MATLAB [8]. They are aimed at providing powerful and reliable tools for solving all kinds of matrix polynomial equations arising in control or signal processing problems.

Some aspects of our work need to be investigated in more detail. First, an interpolation approach to solve the discrete-time symmetric matrix polynomial equation should be derived without extra difficulty. This may prove to be useful when input polynomial matrices are only known through interpolated values, see [1]. Second, an extension of our algorithm to the case when entries of input polynomial matrices are polynomials with complex coefficients seems to be required for signal processing applications [10]. Finally, the special structure of the reduced Sylvester matrix may perhaps be exploited to further reduce computational load. All these aspects are currently subject to further investigation.

Acknowledgments

The first author would like to thank Dr. Jan Ježek for valuable discussions and Prof. Vladimír Kučera, director of the Institute of Information Theory and Automation, whose hospitality and support were much appreciated.

References

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

[2] G. H. Golub and C. F. Van Loan, Matrix Computations - Second Edition, *The Johns Hopkins University Press*, 1989.

[3] D. Henrion and M. Šebek, Symmetric Matrix Polynomial Equation: Interpolation Results, to appear in *Automatica*, 1998.

[4] J. Ježek and V. Kučera, Efficient Algorithm for Matrix Spectral Factorization, *Automatica*, Vol. 21, No. 6, pp. 663–669, 1985.

[5] J. Ježek, Symmetric Matrix Polynomial Equations, *Kybernetika*, Vol. 22, No. 1, pp. 19–30, 1986.

[6] T. Kailath, Linear Systems, *Prentice Hall*, 1980.

[7] V. Kučera, Analysis and Design of Discrete Linear Control Systems, *Prentice Hall International Series in Systems and Control Engineering*, 1991.

[8] H. Kwakernaak and M. Šebek, The Polynomial Toolbox for MATLAB, Version 1.5, 1997. Available as freeware at `http://www.math.utwente.nl/polbox`.

[9] P. Lancaster and M. Tismenetsky, The Theory of Matrices - Second Edition with Applications, *Academic Press*, 1985.

[10] 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.