Technical Report TR2009/240

Multivariable ARMA Systems — Making a Polynomial Matrix Proper

Andrew P. Papliński
Clayton School of Information Technology
Monash University, Clayton 3800, Australia
`Andrew.Paplinski@infotech.monash.edu.au`

May 4, 2009

Abstract

We present two algorithms for polynomial matrices particularly useful in dealing with MIMO ARMA (or left polynomial fraction) models. The first algorithms makes a polynomial matrix proper by adding a required number of zero eigenvalues. The second algorithm extracts a nilpotent part of a polynomial matrix. Both algorithms operate on matrix polynomial coefficients and are easy to implement in MATLAB.

Keywords: Polynomial matrices, Matrix polynomials, polynomial fractions, MIMO dynamic systems, ARMA models.

1 Introduction

Since the seminal works [1, 2] polynomial matrices are widely used in theory and practice of the multi-input multi-output dynamic systems. In this paper we present two algorithms for polynomial matrices particularly useful in dealing with MIMO ARMA (or left polynomial fraction) models. The first algorithms makes a polynomial matrix proper by multiplying it by a properly selected nilpotent polynomial matrix thus adding a required number of zero eigenvalues. The second algorithm factorises a polynomial matrix into a non-nilpotent and nilpotent factors. Both algorithms operate on matrix polynomial coefficients using the QR factorisation and are easy to implement in MATLAB.

The origin of the algorithms presented in this paper can be traced back to [3, 4, 5]. Presentation of this refined version of the algorithm to make a polynomial matrix proper has been induced by the following description of the polyeig function in MATLAB [6] which implies that the algorithm has not yet been absorbed by the researches interested in multiviariable ARMA models.

> [X,e] = polyeig(A0,A1,...Ap) solves the polynomial eigenvalue problem of degree p:
>
> $$(A_0 + \lambda A_1 + \ldots + \lambda^p A_p)x = 0$$
>
> where polynomial degree p is a non-negative integer, and A0, A1, ... Ap are input matrices of order n. Output matrix X, of size n-by-n*p, contains eigenvectors in its columns. Output vector e, of length n*p, contains eigenvalues.
> (...)
> If both A0 and Ap are singular, the problem is potentially ill posed; solutions might not exist or they might not be unique. In this case, the computed solutions may be inaccurate. polyeig attempts to detect this situation and display an appropriate warning message. If either one, but not both, of A0 and Ap is singular, the problem is well posed but some of the eigenvalues may be zero or infinite (Inf).

In particular, we will show that "If both A0 and Ap are singular" the problem is well posed and we will describe an algorithm to be used in such a case. This algorithm can be also used "If either one, but not both, of A0 and Ap is singular".

We hope that the algorithms presented in this paper will be implemented in the next version of the Polynomial Toolbox for MATLAB [7].

2 Problem specification

Consider a matrix polynomial of degree p:

$$A(z) = A_p z^p + \ldots + A_1 z + A_0 \tag{1}$$

where each coefficient A_i is a square $n \times n$ matrix. If the leading coefficient, A_p, is nonsingular the matrix polynomial is called **proper**.

A matrix polynomial can be re-written as a matrix of polynomials, that is, as a polynomial matrix:

$$A(z) = \begin{bmatrix} a_{11}(z) & a_{12}(z) & \cdots \\ a_{21}(z) & a_{22}(z) & \cdots \\ \vdots & \vdots & \end{bmatrix} \tag{2}$$

where each entry is a polynomial of, at most, pth degree: $a_{ij}(z) = a_{ijp}z^p + \ldots + a_{ij0}$. Let the characteristic polynomial of a polynomial matrix:

$$d(z) = \det A(z) = d_m z^m + d_{m-1} z^{m-1} + \ldots + d_0 \tag{3}$$

be of degree m. The matrix $A(z)$ has then typically m eigenvalues. The following properties relating p, n and m are relatively easy to establish [8, 9]:

$$A(z) \text{ is proper} \quad \Leftrightarrow \quad m = n \cdot p \tag{4}$$

$$\operatorname{rank}(A_p) < n \quad \Leftrightarrow \quad m < n \cdot p \tag{5}$$

$$A_0 \text{ is nonsingular} \quad \Leftrightarrow \quad d_0 \neq 0 \tag{6}$$

Now we can formulate two lemmas to address the cases when either the coefficient A_p, or A_0 of a polynomial matrix $A(z)$ is singular.

Lemma 1 *Making a polynomial matrix proper (eqns (4) and (5)).*
If

- *$A(z)$ as in eqn (1) has a singular leading coefficient, A_p, and*
- *$Q(z)$ is a nilpotent polynomial matrix with m_q zero eigenvalues, that is,*

$$\det Q(z) = z^{m_q} \qquad \textit{where} \qquad m + m_q = n \tag{7}$$

then

$$\hat{A}(z) = Q(z) \cdot A(z) \tag{8}$$

is proper (has a nonsingular leading coefficient, $\hat{A}_p$).

Similarly, we can formulate the following **factorisation** lemma for the case of A_0 being singular, that is, the $A(z)$ having some zero eigenvalues.

Lemma 2 *Extracting a non-nilpotent part of a polynomial matrix (eqn (6)).*
If

- *A_0 is singular, and*
- *the characteristic polynomial (eqn (3)) has its m_q least significant coefficients being zeros,*

$$d_0 = \ldots d_{m_q-1} = 0 \ , \quad d_{m_q} \neq 0$$

then it is possible to factorize $A(z)$ so that

$$A(z) = Q(z) \cdot \hat{A}(z) \tag{9}$$

where $Q(z)$ is a nilpotent polynomial matrix such that

$$\det Q(z) = z^{m_q} \tag{10}$$

and $\hat{A}(z)$ is a non-nilpotent polynomial matrix, that is, having only nonzero eigenvalues.

In sec. 4, we are going to prove Lemma 1 by construction. As far as Lemma 2 is concerned, it is possible to note that a polynomial matrix $A(z)$ with a singular least significant coefficient, A_0, can be replaced by a matrix $A(z^{-1})$ in which A_0 is moved to the most significant position, as in Lemma 1. We will elaborate on it in sec. 4

3 Multivariable ARMA Model Interpretation

Consider a discrete-time dynamic system with n output signals, $y(k)$, and n_i input signals, $u(k)$, described by the following ARMA (left polynomial fraction) model in the operator form

$$A(z)y(k) = B(z)u(k) \tag{11}$$

where the output matrix, $A(z)$, is as in eqn (1), and the input matrix, $B(z)$, is a $n \times n_i$ polynomial matrix of degree $p-1$. The characteristic polynomial of the dynamic system is given by eqn (3), hence, the system order is m. After multiplication by z^{-p} eqn (11) can be first rewritten as,

$$(A_p + A_{p-1}z^{-1} + \ldots + A_0 z^{-p})y(k) = (B_{p-1}z^{-1} + \ldots + B_0 z^{-p})u(k) \tag{12}$$

and subsequently as

$$A_p y(k) = -\begin{bmatrix} A_{p-1} & \ldots & A_0 \end{bmatrix} \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-p) \end{bmatrix} + \begin{bmatrix} B_{p-1} & \ldots & B_0 \end{bmatrix} \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-p) \end{bmatrix} \tag{13}$$

It is easy to note that in order to obtain the modelling ARMA form in which the current value of the output signals, $y(k)$, is a matrix linear combination of the past output signals, $y(k-i)$, and input signals, $u(k-i)$, the leading coefficient, A_p must be nonsingular, that is, the output matrix, $A(z)$, must be proper. However, each time when the system order, m, the number of outputs, n, and the maximum system delay, p, satisfy eqn (5), the leading coefficient A_p is singular. In such a case, applying the results of Lemma 1, the dynamic system 11 can be modified by premultiplication by a nilpotent matrix $Q(z$ as in eqn (8) which yields the following

$$Q(z)A(z)y(k) = Q(z)B(z)u(k)$$

or, finally

$$\hat{A}(z)y(k) = \hat{B}(z)u(k) \quad \text{where} \quad \hat{B}(z) = Q(z)B(z) \tag{14}$$

Now the leading coefficient, $\hat{A}_p$ is nonsingular, and the ARMA model as in eqn (13) can be effectively formed.

4 The Algorithms

Before we formulate the algorithm to find $Q(z)$ which can be used to convert a polynomial matrix into a proper form, as in eqn (8), we first introduce convenient building blocks of the algorithm. Consider a family of elementary nilpotent $n \times n$ first degree polynomial matrices in the following form:

$$U_k(z) = U_{k0}z + U_{k1} = \begin{bmatrix} \mathbf{0} & I_r \\ zI_k & \mathbf{0} \end{bmatrix}, \quad k + r = n \tag{15}$$

Note that

$$\det U_k(z) = \pm z^k \tag{16}$$

The block column matrix of coefficients of $U_k(z)$ is of the size $2n \times n$:

$$U_k = \begin{bmatrix} U_{k0} \\ U_{k1} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_r \\ I_n \\ \mathbf{0}_k \end{bmatrix} \tag{17}$$

Matrices $U_k(z)$ have the shift-up property, that is, $U_k(z) \cdot A(z)$ shifts rows of the block-column matrix of coefficients of $A(z)$ up by k positions as presented in Figure 1.

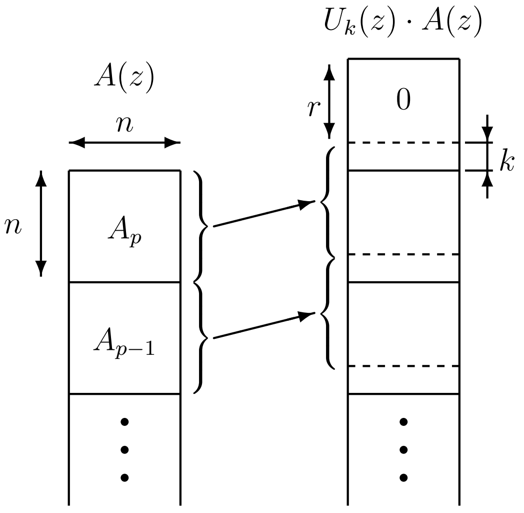

Figure 1: A matrix $U_k(z)$ as a shift-up operator

4.1 Making a Polynomial Matrix Proper

Using the above matrices, the algorithm, based on that presented in [3, 4, 5], to find a nilpotent $Q(z)$ which makes, $A(z)$ proper as in eqn (8) can be formulated as in Figure 2. This algorithm can be conveniently written in MATLAB.

The algorithm calculates $\hat{A}(z)$ and $Q(z)$ as in Lemma 1, and operates on the coefficients of the polynomial matrix, $A(z)$, arranged in a block-column matrix, the leading coefficient, A_p, occupying the top position.

At each ith step of the algorithm[1], using the QR factorisation, the current leading coefficient, A_p, is expanded into a product of a unitary matrix, P_i, and a triangular matrix, R_i. If the rank of A_p is k_i, then the triangular matrix R_i has k_i zero rows, as presented in Figure 2. Using a shift-up nilpotent polynomial matrix $U_{k_i}(z)$, the block-column matrix of coefficients is shifted up by k_i positions, and if the new A_p is still singular, the algorithm step is repeated. Finally, the $Q(z)$ matrix can be expressed as a product of elementary nilpotent

[1]For the sake of simplicity, in the flow-chart of Figure 2, the successive values of $\hat{A}(z)$ are marked $A(z)$, and the subscript i has been dropped out.

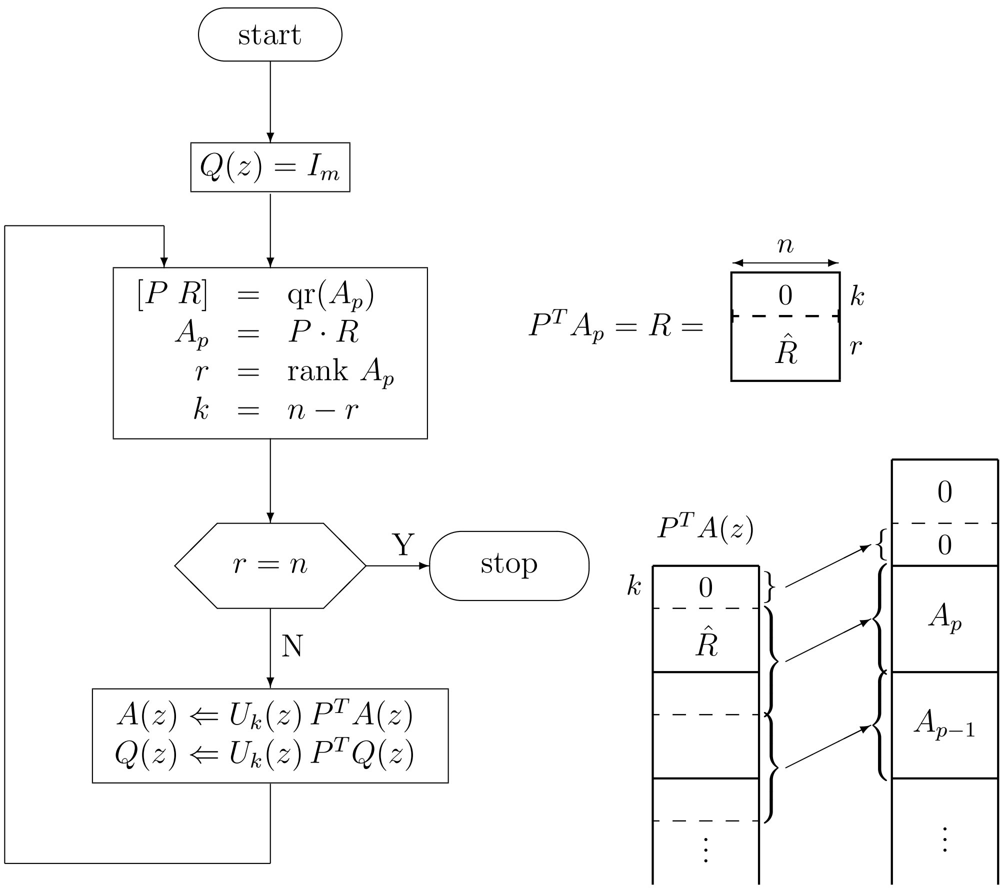

Figure 2: The algorithm

matrices, $U_{k_i}(z)$ and unitary $n \times n$ matrices, P_i, obtained during the QR factorisation, that is,

$$Q(z) = \prod_i U_{k_i}(z) \cdot P_i^T \tag{18}$$

Eqn (18) is a constructive proof of the Lemma 1.

4.2 Extracting a Non-Nilpotent Factor

The algorithm presented in Figure 2 can be also used to extract the non-nilpotent part of a polynomial matrix as in Lemma 2 (eqn (9)).

Let us assume that the least significant coefficient, A_0, is singular, which implies the presence of zero eigenvalues.

We first observe that the following matrix polynomial

$$\tilde{A}(z^{-1}) = z^{-p} A(z) \tag{19}$$

have the same coefficients as $A(z)$ only arranged in the reversed order. The leading coefficient of $\tilde{A}(z^{-1})$ is now A_0. If A_0 is singular, we can now apply the algorithm of Figure 2 to the polynomial matrix, $\tilde{A}(z^{-1})$ so that we obtain the following proper polynomial matrix:

$$\hat{\tilde{A}}(z^{-1}) = Q(z^{-1}) \cdot \tilde{A}(z^{-1}) \tag{20}$$

Multiplication of eqn. (20) by $z^p \cdot Q^{-1}(z^{-1})$ yields the following factorization:

$$A(z) = \tilde{Q}(z) \cdot \hat{A}(z) \tag{21}$$

where

$$\hat{A}(z) = z^p \cdot \hat{\tilde{A}}(z^{-1}) \tag{22}$$

is the non-nilpotent factor representing a non-zero eigenvalue part of the dynamic system (11) and $Q(z)$ is the nilpotent factor

$$\tilde{Q}(z) = Q^{-1}(z^{-1}) = P_i \cdot \prod_i U_{k_i}^{-1}(z^{-1}) \tag{23}$$

where the elementary shift matrices are of the following form

$$U_k^{-1}(z^{-1}) = V_k(z) = \begin{bmatrix} \mathbf{0} & zI_k \\ I_r & \mathbf{0} \end{bmatrix} \tag{24}$$

Concluding Remarks

The two algorithms presented above aim at making either the most or the least significant polynomial matrix coefficient non-singular. This is important specifically when the Sylvester matrices approach is utilised [5, 10].

References

[1] T. Kailath, *Linear Systems.* New York: Englewood Cliffs, 1980.

[2] F. M. Callier and C. A. Desoer, *Multivariabe Feedback Systems.* New York: Springer-Verlag, 1982.

[3] M. W. Rogoziński, A. P. Papliński, and M. J. Gibbard, "An algorithm for the calculation of a nilpotent interactor matrix for linear multivariable systems," *IEEE Transactions on Automatic Control*, vol. AC–32, pp. 234–237, March 1987.

[4] A. P. Papliński and M. W. Rogoziński, "Right nilpotent interactor matrix and its application to multivariable stochastic control," in *Proceedings of the 1990 American Control Conference, San Diego*, May 1990.

[5] P. Stefanidis, A. P. Papliński, and M. J. Gibbard, *Numerical Operations with Polynomial Matrices — Application to Multi-Variable Dynamic Compensator Design.* Springer-Verlag, 1992.

[6] *MATLAB Function Reference.* Natick, MA 01760-1500, USA: The MathWorks, Inc., 1999. Version 5.

[7] M. Šebek, H. Kwakernaak, D. Henrion, and S. Peichová, "Recent progress in polynomial methods and polynomial toolbox for MATLAB version 2.0," in *Proceedings of the 37th IEEE Conference on Decision & Control*, vol. 3979, (Tampa, Florida, USA), December 1998.

[8] S. Barnett, *Polynomials and Linear Control Systems.* New York and Basel: Marcel Dekker, 1983.

[9] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials.* London: Academic Press, 1982.

[10] 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.