Solution of Discrete ARMA-Representations via MAPLE.

J. Jones[†], N. P. Karampetakis[‡] and A. C. Pugh[†]

[†]*Department of Mathematical Sciences*
Loughborough University of Technology
Loughborough, Leics
ENGLAND, U.K.

[‡]*Department of Mathematics*
Aristotle University of Thessaloniki
54006 *GREECE*

Abstract

In [3] closed formulae for the forward, backward and symmetric solutions of an ARMA–Representation have been presented. Here these formulae are implemented in the symbolic computational language MAPLE and corresponding MAPLE code is provided.

1 Introduction

Consider the regular, discrete–time, AutoRegressive Moving Average (ARMA)–Representation described by the matrix equation

$$A(\sigma)y(k) = B(\sigma)u(k) \tag{1.1a}$$

where

$$\begin{aligned} A(\sigma) &= A_0 + A_1\sigma + \ldots + A_q\sigma^q \in \mathbb{R}[\sigma]^{r\times r}\ , \quad \text{rank } A(\sigma) = r \\ B(\sigma) &= B_0 + B_1\sigma + \ldots + B_q\sigma^q \in \mathbb{R}[\sigma]^{r\times m} \end{aligned} \tag{1.1b}$$

where at least one of A_q, B_q is non–zero, σ denotes the advance operator (i.e. $\sigma^i y(k) = y(k+i)$), $y(k) : \mathbb{Z}^+ \to \mathbb{R}^r$ defines the output and $u(k) : \mathbb{Z}^+ \to \mathbb{R}^m$ defines the input of the system (1.1).

In the case where $A(\sigma) = \sigma E - A \in \mathbb{R}[\sigma]^{r\times r}$ and $B(\sigma) = B \in \mathbb{R}^{r\times m}$ the ARMA–Representation (1.1) reduces to the Generalized State Space(GSS)–Representation

$$Ey(k+1) = Ay(k) + Bu(k) \tag{1.2}$$

Further if E is non–singular (i.e. $|E| \neq 0$) then (1.2) reduces to the State Space(SS)–Representation.

In [3] closed formulae for the forward, backward and symmetric solutions of the ARMA–Representation (1.1) are presented. These formulae depend explicitly on the *forward fundamental matrix* H_k and the *backward fundamental matrix* V_k of $A(z)$ both of which can be seen to be of fundamental importance in the analysis of discrete ARMA–Representations. The closed formulae obtained can be seen to be natural extensions to the ones proposed by [4] for the GSS–Representation (1.2) which correspondingly depend explicitly on the *forward fundamental matrix* ϕ_k and the *backward fundamental matrix* τ_k of $(zE - A)$.

In this paper we implement the closed formulae as given in [3] for the forward, backward and symmetric solutions of the ARMA-Representation (1.1) in the symbolic computational language MAPLE [1]. In section 2 the definition and construction of H_k and V_k will be considered. In section 3 the closed formulae for the solution of the ARMA–Representation (1.1) will be shown and these will be implemented via MAPLE in section 4 where the corresponding MAPLE procedures will be presented. Finally in section 5 these procedures will be illustrated via a numerical example.

2 Preliminaries

Consider the ARMA–Representation (1.1) and write it in the expanded form

$$A_q y(k+q) + \cdots + A_0 y(k) = B_q u(k+q) + \cdots B_0 u(k) \tag{2.1}$$

where we assume $A(z)$ is *regular* (i.e. $\det[A(z)] \neq 0$), $k \in [0,\ N]$ and $u(k)$ is non–zero for $k = 0, 1, \ldots, N-q$. For such as case we can define the following

Definition 2.1 (*Forward Fundamental Matrix Sequence of* $A(z)$) Let the Laurent series expansion of $A(z)^{-1}$ about $z = \infty$ be given by

$$A(z)^{-1} = H_{\hat{q}_r} z^{\hat{q}_r} + H_{\hat{q}_r - 1} z^{\hat{q}_r - 1} + \cdots + H_1 z + H_0 + H_{-1} z^{-1} + \cdots \tag{2.2}$$

where $\hat{q}_r$ is the greatest order zero of $A(z)$ at $s = \infty$. Then the coefficient matrices $H_j \in \mathbb{R}^{r \times r},\ j \leq \hat{q}_r$ constitute the *forward fundamental matrix sequence* of the matrix $A(z)$. □

Definition 2.2 (*Backward Fundamental Matrix Sequence of* $A(z)$) Let the Laurent series expansion of $A(z)^{-1}$ about $z = 0$ be given by

$$A(z)^{-1} = V_{-l} z^{-l} + V_{-l+1} z^{-l+1} + \cdots + V_{-1} z^{-1} + V_0 + V_1 z + \cdots \tag{2.3}$$

Then the coefficient matrices $V_j \in \mathbb{R}^{r \times r},\ j \geq -l$ constitute the *backward fundamental matrix sequence* of the matrix $A(z)$. □

To compute the forward fundamental matrix H_k of $A(z)$ an algorithm by [2] has been proposed. This is based on a recursive algorithm for the computation of the inverse of a polynomial matrix $A(z)$ in terms of only its determinant and the coefficients of its adjoint matrix. (This is in turn a generalization of a Leverrier–type algorithm presented in [5] for the computation of the inverse of the pencil $(zE - A)$). This algorithm can also be used to compute the backward fundamental matrix V_k of $A(z)$ as was noted in [3]. Here it was shown that V_k can be found directly from examining the dual polynomial matrix $\tilde{A}(w)$ of $A(z)$ defined as

$$\tilde{A}(w) = A_0 w^q + A_1 w^{q-1} + \cdots + A_{q-1} w + A_q \tag{2.4}$$

and its corresponding forward fundamental matrix $\tilde{H}_k$ where

$$\tilde{A}(w)^{-1} = \tilde{H}_f w^f + \tilde{H}_{f-1} w^{f-1} + \cdots + \tilde{H}_1 w + \tilde{H}_0 + \tilde{H}_{-1} w^{-1} + \cdots \tag{2.5}$$

Using this approach enables the backward fundamental matrix V_k to be obtained.

Alternatively H_k (resp. V_k) can be found directly using the existing MAPLE procedure `series` via an expression such as

```
>map(series,inverse(A),z=infinity,5);
```

(`>map(series,inverse(A),z=0,5);`)

where `A` denotes the polynomial matrix $A(z)$ and the last figure denotes the order of the expansion required. Such an expression expands each element in the matrix `inverse(A)` as a Laurent expansion about the point $z = \infty$ (resp. $z = 0$). Therefore the coefficient matrices H_j (resp. V_j) as defined above can be directly obtained by examining the coefficients of each such element for each degree of z. This can be achieved for example via a short procedure `MATEXP` which is described in section 4.

The procedures presented in section 4 for the solution of the ARMA–Representation (1.1) constructs the matrices H_k (resp. V_k) using this latter method. It is envisaged to implement the algorithm of [2] via MAPLE in the very near future due to it being a more efficient way of inverting a polynomial matrix.

3 Solution of ARMA–Representations

In [3] closed formulae for the forward, backward and symmetric solutions of the ARMA–Representation (1.1) were obtained. In this section each of these three types of solution are considered and the corresponding closed formulae provided.

3.1 Forward Solution

Given a set of admissible initial conditions, namely $\{y(0),\ y(1),\ \ldots,\ y(q-1)\}$, our aim is to determine $y(k),\ (k = q,\ q+1,\ \ldots)$, in a *forward* fashion from the input sequence $u(k)$ and *previous* values of the output. From [3] we have the following result.

Theorem 3.1 (*Forward Solution*) Consider the ARMA–Representation as in (1.1) where $A(z)$ is regular and let the Laurent series expansion of $A(z)^{-1}$ about the point $z = \infty$ be as given in (2.2). Then the forward response of the system for $k \geq q$ is given by

$$
\begin{aligned}
y(k) = \ & \begin{bmatrix} H_{-k-q} & H_{-k-q+1} & \ldots & H_{-k-1} \end{bmatrix} \begin{pmatrix} A_q & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_1 & \cdots & A_q \end{pmatrix} \begin{pmatrix} y(0) \\ \vdots \\ y(q-1) \end{pmatrix} \\
& + \begin{bmatrix} H_{-k} & H_{-k+1} & \ldots & H_{\hat{q}_r} \end{bmatrix} \begin{pmatrix} B_0 & \cdots & B_q & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_0 & \cdots & B_q \end{pmatrix} \begin{pmatrix} u(0) \\ u(1) \\ \vdots \\ u(k+\hat{q}_r+q) \end{pmatrix}
\end{aligned} \tag{3.1}
$$

□

It is clear that for $k = 0,\ 1,\ \ldots,\ q-1$ we cannot use (3.1) to determine $y(0),\ y(1),\ \ldots,\ y(q-1)$. In general these initial conditions cannot be arbitrarily chosen and must *satisfy* (3.1) for $k = 0,\ 1,\ \ldots,\ q-1$ for the ARMA–Representation (1.1) to have a solution. Therefore from [3] we have

Theorem 3.2 (*Forward Compatibility Condition*) The ARMA–Representation (1.1) has a solution iff the compatibility condition

$$\tilde{\mathcal{A}}_1 \begin{pmatrix} H_0 & \cdots & H_{q-1} \\ \vdots & \ddots & \vdots \\ H_{-q+1} & \cdots & H_0 \end{pmatrix} \tilde{\mathcal{A}}_1 \begin{pmatrix} y(0) \\ \vdots \\ y(q-1) \end{pmatrix} =$$

$$\tilde{\mathcal{A}}_1 \begin{pmatrix} H_0 & \cdots & H_{\hat{q}_r} & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ H_{-q+1} & \cdots & H_{\hat{q}_r-q+1} & \cdots & H_{\hat{q}_r} \end{pmatrix} \begin{pmatrix} B_0 & \cdots & B_q & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_0 & \cdots & B_q \end{pmatrix} \begin{pmatrix} u(0) \\ u(1) \\ \vdots \\ u(2q+\hat{q}_r-1) \end{pmatrix} \Bigg]$$

where

$$\tilde{\mathcal{A}}_1 = \begin{pmatrix} A_0 & \cdots & A_{q-1} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_0 \end{pmatrix} \tag{3.2}$$

is satisfied for the initial condition vectors $\{y(0),\ y(1),\ \ldots,\ y(q-1)\}$ under non–zero inputs. □

3.2 Backward Solution

Given a set of admissible final conditions, namely $\{y(N),\ y(N-1),\ \ldots,\ y(N-q+1)\}$, our aim is to determine $y(k),\ (k = N-q,\ N-q-1,\ \ldots)$, in a *backwards* fashion from the input sequence $u(k)$ and *future* values of the output. Again from [3] we have the following result.

Theorem 3.3 (*Backward Solution*) Consider the ARMA–Representation as in (1.1) where $A(z)$ is regular and let the Laurent series expansion of $A(z)^{-1}$ about the point $z = 0$ be as given in (2.3). Then the backward response of the system for $k \leq N-q$ is given by

$$\begin{aligned} y(k) = \ & \begin{bmatrix} V_{N-k} & V_{N-k-1} & \ldots & V_{N-k-q+1} \end{bmatrix} \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} y(N) \\ \vdots \\ y(N-q+1) \end{pmatrix} \\ & + \begin{bmatrix} V_{N-k-q} & V_{N-k-q-1} & \ldots & V_{-l} \end{bmatrix} \begin{pmatrix} B_q & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_0 \end{pmatrix} \begin{pmatrix} u(N) \\ u(N-1) \\ \vdots \\ u(k-l) \end{pmatrix} \end{aligned} \tag{3.3}$$

□

Analogous to the forward solution it is clear that for $k = N,\ N-1,\ \ldots,\ N-q+1$ we cannot use (3.3) to determine $y(N),\ y(N-1),\ \ldots,\ y(N-q+1)$. In general these final conditions cannot be arbitrarily chosen and must *satisfy* (3.3) for $k = N,\ N-1,\ \ldots,\ N-q+1$ for the ARMA–Representation (1.1) to have a solution. Therefore again from [3] we have

Theorem 3.2 (*Backward Compatibility Condition*) The ARMA–Representation (1.1) has a solution iff the compatibility condition

$$\tilde{\mathcal{A}}_2 \begin{pmatrix} V_{-q} & \cdots & V_{-2q+1} \\ \vdots & \ddots & \vdots \\ V_{-1} & \cdots & V_{-q} \end{pmatrix} \tilde{\mathcal{A}}_2 \begin{pmatrix} y(N) \\ \vdots \\ y(N-q+1) \end{pmatrix} =$$

$$\tilde{\mathcal{A}}_2 \begin{pmatrix} V_{-q} & \cdots & V_{-l} & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ V_{-1} & \cdots & V_{-l+q-1} & \cdots & V_{-l} \end{pmatrix} \begin{pmatrix} B_q & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_0 \end{pmatrix} \begin{pmatrix} u(N) \\ u(N-1) \\ \vdots \\ u(N-q-l+1) \end{pmatrix} \Bigg]$$

where

$$\tilde{\mathcal{A}}_2 = \begin{pmatrix} A_q & \cdots & A_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_q \end{pmatrix} \tag{3.4}$$

is satisfied for the final condition vectors $\{y(N),\ y(N-1),\ \ldots,\ y(N-q+1)\}$ under non–zero inputs. □

3.3 Symmetric Solution

Consider (1.1) as a relation between the inputs $u(k)$ and the outputs $y(k)$ over an interval $[0,\ N]$. Then given a set of admissible initial and final conditions, namely $\{y(0),\ y(1),\ \ldots,\ y(q-1)\}$ and $\{y(N),\ y(N-1),\ \ldots,\ y(N-q+1)\}$, our aim is to determine $y(k)$ for $k \in [0,\ N]$ from the input sequence $u(k)$ and these *initial* and *final* output conditions. From [3] we have the following result.

Theorem 3.5 (*Symmetric Solution*) Consider the ARMA–Representation as in (1.1) where $A(z)$ is regular and let the Laurent series expansion of $A(z)^{-1}$ about the point $z = \infty$ be as given in (2.2). Then the symmetric response of the system for $q \le k \le N - q$ is given by

$$\begin{aligned} y(k) = \ & \begin{bmatrix} H_{N-k} & \cdots & H_{N-k-q+1} \end{bmatrix} \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} y(N) \\ \vdots \\ y(N-q+1) \end{pmatrix} \\ & + \begin{bmatrix} H_{-k-1} & \cdots & H_{-k-q} \end{bmatrix} \begin{pmatrix} A_q & \cdots & A_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_q \end{pmatrix} \begin{pmatrix} y(q-1) \\ \vdots \\ y(0) \end{pmatrix} \qquad (3.5) \\ & + \begin{bmatrix} H_{N-k-q} & \cdots & H_{-k} \end{bmatrix} \begin{pmatrix} B_q & B_{q-1} & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & B_q & \cdots & B_1 & B_0 & & \vdots \\ \vdots & & \ddots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_1 & B_0 \end{pmatrix} \begin{pmatrix} u(N) \\ \vdots \\ \vdots \\ u(0) \end{pmatrix} \end{aligned}$$

□

As in the forward and backward solutions there exists restrictions such that the ARMA–Representation (1.1) has a solution. These restrictions exist between the input sequence $u(k)$ and the initial and final output conditions and form the *boundary mapping equation* of (1.1). Again from [3] we have

Theorem 3.6 (*Boundary Mapping Equation*) The ARMA–Representation (1.1) has a solution iff the *boundary mapping equation*

$$\left(\begin{array}{ccc|ccc} H_{-q} & \cdots & H_{-2q+1} & H_{-N+q-1} & \cdots & H_{-N} \\ \vdots & & \vdots & \vdots & & \vdots \\ H_{-1} & \cdots & H_{-q} & H_{-N+2q-2} & \cdots & H_{-N+q-1} \\ \hline H_{N-2q+1} & \cdots & H_{N-3q+2} & H_0 & \cdots & H_{-q+1} \\ \vdots & & \vdots & \vdots & & \vdots \\ H_{N-q} & \cdots & H_{N-2q+1} & H_{q-1} & \cdots & H_0 \end{array}\right) \times$$

$$\times \left(\begin{array}{ccc|ccc} A_q & \cdots & A_1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & A_q & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & A_0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & A_{q-1} & \cdots & A_0 \end{array}\right) \left(\begin{array}{c} y(N) \\ \vdots \\ y(N-q+1) \\ \hline y(q-1) \\ \vdots \\ y(0) \end{array}\right) = \tag{3.6}$$

$$\left(\begin{array}{ccc} H_{-q} & \cdots & H_{-N} \\ \vdots & & \vdots \\ H_{-1} & \cdots & H_{-N+q-1} \\ \hline H_{N-2q+1} & \cdots & H_{-q+1} \\ \vdots & & \vdots \\ H_{N-q} & \cdots & H_0 \end{array}\right) \left(\begin{array}{ccccccc} B_q & B_{q-1} & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & B_q & \cdots & B_1 & B_0 & & \vdots \\ \vdots & & \ddots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_1 & B_0 \end{array}\right) \left(\begin{array}{c} u(N) \\ \vdots \\ \vdots \\ u(0) \end{array}\right)$$

is satisfied for the input sequence $u(k)$ and the initial and final condition vectors $\{y(0),\ y(1),\ \ldots,\ y(q-1)\}$ and $\{y(N),\ y(N-1),\ \ldots,\ y(N-q+1)\}$. □

3.4 Alternative Solution

Consider the forward solution as in (3.1). An obvious disadvantage of (3.1) is that for each successive output $y(k)$, specified by $k = q,\ q+1,\ \ldots$, the coefficient matrices H_j comprising each specific solution change. Therefore if the solution is required over a comparatively large range, say $[y(2),\ y(3),\ \ldots,\ y(100)]$ corresponding to $k = 2,\ 3,\ \ldots,\ 100$, we would require the coefficient matrices $H_{-101},\ H_{-100},\ \ldots,\ H_{\hat{q}_r}$. In [3] an equivalent forward solution was presented which for the general solution $y(k)$ depends on the *previous* q outputs $\{y(k-1),\ y(k-2),\ \ldots,\ y(k-q)\}$ and not on the q fixed initial conditions $\{y(0),\ y(1),\ \ldots,\ y(q-1)\}$. In this case the coefficient matrices required over a solution range is fixed, (i.e. independent of k), namely $H_{-q},\ H_{-q+1},\ \ldots,\ H_{\hat{q}_r}$.

Similarly the same approach can be applied to the backward solution as given in (3.3) where the equivalent solution depends on the *future* q outputs

$\{y(k+1),\ y(k+2),\ \ldots,\ y(k+q)\}$ and not on the q fixed final conditions $\{y(N),\ y(N-1),\ \ldots,\ y(N-q+1)\}$. In this case the coefficient matrices required over a solution range is again fixed, (i.e. independent of k), namely $V_{-l},\ V_{-q+1},\ \ldots,\ V_q$.

Theorem 3.7 (*Alternative Forward and Backward Solutions*) The forward and backward solutions to the ARMA–Representation (1.1) as given in (3.1) and (3.3) can be written in the following alternative forms

FORWARD SOLUTION

$$\begin{aligned} y(k) = \; & -\begin{bmatrix} H_{-1} & \cdots & H_{-q} \end{bmatrix} \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} y(k-1) \\ \vdots \\ y(k-q) \end{pmatrix} \\ & + \begin{bmatrix} H_{-q} & H_{-q+1} & \ldots & H_{\hat{q}_r} \end{bmatrix} \begin{pmatrix} B_0 & \cdots & B_q & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_0 & \cdots & B_q \end{pmatrix} \begin{pmatrix} u(k-q) \\ u(k-q+1) \\ \vdots \\ \vdots \\ u(k+\hat{q}_r+q) \end{pmatrix} \end{aligned} \tag{3.7}$$

BACKWARD SOLUTION

$$\begin{aligned} y(k) = \; & \begin{bmatrix} V_q & \cdots & V_1 \end{bmatrix} \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} y(k+q) \\ \vdots \\ y(k+1) \end{pmatrix} \\ & + \begin{bmatrix} V_0 & V_{-1} & \ldots & V_{-l} \end{bmatrix} \begin{pmatrix} B_q & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_0 \end{pmatrix} \begin{pmatrix} u(k+q) \\ u(k+q-1) \\ \vdots \\ u(k-l) \end{pmatrix} \end{aligned} \tag{3.8}$$

□

The alternative forward and backward solutions (3.7) and (3.8) can be seen to be equivalent to the corresponding solutions (3.1) and (3.3) for $k = q$ and $k = N - q$ respectively. Clearly the compatibility conditions (3.2) and (3.4) hold for the respective solutions (3.7) and (3.8) as they still define an admissible set of initial and final conditions, under a given non–zero input sequence, for a solution to exist.

For the case of the symmetric solution as given in (3.5) we can form two alternative solutions. These are termed the *forward–symmetric* and *backward–symmetric* solutions of the ARMA–Representation (1.1) as presented in [3].

Theorem 3.8 (*Forward–Symmetric and Backward–Symmetric Solutions*) The symmetric solution (3.5) can be written in the alternative forms

FORWARD–SYMMETRIC

$$
\begin{aligned}
y(k) = \; & \begin{bmatrix} H_{N-k} & \cdots & H_{N-k-q+1} \end{bmatrix} \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} y(N) \\ \vdots \\ y(N-q+1) \end{pmatrix} \\
& - \begin{bmatrix} H_{-1} & \cdots & H_{-q} \end{bmatrix} \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} y(k-1) \\ \vdots \\ y(k-q) \end{pmatrix} \qquad (3.9) \\
& + \begin{bmatrix} H_{N-k-q} & \cdots & H_{-q} \end{bmatrix} \begin{pmatrix} B_q & B_{q-1} & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & B_q & \cdots & B_1 & B_0 & & \vdots \\ \vdots & & \ddots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_1 & B_0 \end{pmatrix} \begin{pmatrix} u(N) \\ \vdots \\ \vdots \\ u(k-q) \end{pmatrix}
\end{aligned}
$$

BACKWARD–SYMMETRIC

$$
\begin{aligned}
y(k) = \; & - \begin{bmatrix} H_0 & \cdots & H_{-q+1} \end{bmatrix} \begin{pmatrix} A_q & \cdots & A_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_q \end{pmatrix} \begin{pmatrix} y(k+q) \\ \vdots \\ y(k+1) \end{pmatrix} \\
& + \begin{bmatrix} H_{-k-1} & \cdots & H_{-k-q} \end{bmatrix} \begin{pmatrix} A_q & \cdots & A_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_q \end{pmatrix} \begin{pmatrix} y(q-1) \\ \vdots \\ y(0) \end{pmatrix} \qquad (3.10) \\
& + \begin{bmatrix} H_0 & \cdots & H_{-k} \end{bmatrix} \begin{pmatrix} B_q & B_{q-1} & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & B_q & \cdots & B_1 & B_0 & & \vdots \\ \vdots & & \ddots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_1 & B_0 \end{pmatrix} \begin{pmatrix} u(k+q) \\ \vdots \\ \vdots \\ u(0) \end{pmatrix}
\end{aligned}
$$

□

In the *Forward–Symmetric* case we still solve within the region $[0,\ N]$ but now the solution depends on the q final conditions $\{y(N),\ y(N-1),\ \ldots,\ y(N-q+1)\}$ and the *previous* q outputs $\{y(k-1),\ y(k-2),\ \ldots,\ y(k-q)\}$ and no longer on the q fixed initial conditions $\{y(0),\ y(1),\ \ldots,\ y(q-1)\}$. Therefore we solve *forwards* in the interval.

In the *Backward–Symmetric* case we again still solve within the region $[0,\ N]$ but now the solution depends on the q initial conditions $\{y(0),\ y(1),\ \ldots,\ y(q-1)\}$ and the *future* q outputs $\{y(k+1),\ y(k+2),\ \ldots,\ y(k+q)\}$ and no longer on the q fixed final conditions $\{y(N),\ y(N-1),\ \ldots,\ y(N-q+1)\}$. Therefore we solve *backwards* in the interval.

4 Implementation via MAPLE

In this section we implement the formulae for the solution of the ARMA–Representation (1.1), as presented in section 3, in the symbolic computational language MAPLE [1]. One obvious advantage of using MAPLE is that it enables the user to implement any one of the "built in" procedures inherent in it predominantly, in this case, those in the linear algebra package `linalg` which contains a collection of procedures for matrix manipulation. This results in further simplification of any program code required.

For each solution we will initially give a pseudocode for the corresponding MAPLE procedure. In this pseudocode the following notation will be used.

i) $0_{(m,n)} \stackrel{\text{def}}{=}$ the $(m \times n)$ zero matrix.

ii) $A \cdot B \stackrel{\text{def}}{=}$ the product of the matrices A and B.

iii) $a \leftarrow\!\!\stackrel{\text{def}}{=} a$ is 'assigned to'.

4.1 Forward Solution

Here we implement the alternative forward solution of the ARMA–Representation (1.1) as given in (3.7).

4.1.1 Forward Solution – Pseudocode

INPUT

– The matrices $A(z)$ and $B(z)$

– The integer $q \stackrel{\text{def}}{=}$ highest degree element in $A(z)$ or $B(z)$

PROCEDURE

$Aseries \leftarrow$ the matrix expansion $\{A_0,\ A_1,\ \ldots,\ A_q\}$

$Bseries \leftarrow$ the matrix expansion $\{B_0,\ B_1,\ \ldots,\ B_q\}$

$IA \leftarrow$ the inverse of A

$IAS \leftarrow$ the matrix IA with each term expanded about $z = \infty$

$Hseries \leftarrow$ the matrix expansion $\{H_{-q},\ H_{-q+1},\ \ldots,\ H_{\hat{q}_r}\}$

$$termA \leftarrow \text{the block matrix } \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix}$$

$$termB \leftarrow \text{the block matrix } \begin{pmatrix} B_0 & \cdots & B_q & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_0 & \cdots & B_q \end{pmatrix}$$

$INHone \leftarrow$ the block matrix $[H_{-1}\ H_{-2}\ \cdots\ H_{-q}]$

$INHtwo \leftarrow$ the block matrix $[H_{-q}\ H_{-q+1}\ \cdots\ H_{\hat{q}_r}]$

$$YY \leftarrow \text{the vector} \begin{pmatrix} y(k-1) \\ \vdots \\ y(k-q) \end{pmatrix}$$

$$UU \leftarrow \text{the vector} \left(\begin{array}{c} u(k-q) \\ \vdots \\ u(k) \\ \hline u(k+1) \\ \vdots \\ u(k+\hat{q}_r+q) \end{array}\right)$$

OUTPUT

– The vector $(-1)\cdot[INHone\cdot termA\cdot YY]+[INHtwo\cdot termB\cdot UU]$

4.1.2 Forward Solution – Procedures

For the implementation of (3.7) in MAPLE three procedures are presented; firstly the main calling procedure `FORWARD` and secondly the two sub–procedures `MATEXP` and `DIFFPOW`. These are described below

`FORWARD` – **Parameters**
- `A,B` (*matrices*) Represent polynomial matrices $A(z)$ and $B(z)$.
- `q` (*integer*) The highest degree element in $A(z)$ and $B(z)$.

– **Purpose**
- Computes the forward solution $y(k)$ of the ARMA–Representation $A(\sigma)y(k) = B(\sigma)u(k)$ in terms of the previous q vectors $y(k-1)$, $\cdots$, $y(k-q)$ and the input sequence $u(k)$.

`MATEXP` – **Parameters**
- `A` (*matrix*) Represents a rational matrix $A(z)$ whose elements are in the form of a Laurent expansion in z.
- `q` (*integer*) Initial degree of z considered.
- `st` (*integer*) Final degree of z considered.
- `n` (*integer*) Row dimension of $A(z)$.
- `m` (*integer*) Column dimension of $A(z)$.

– **Purpose**
- Forms the matrix expansion of a matrix $A(z)$ from z^q to z^{st} where $q \geq st$. Output is in a list of the form $[A_{st},\ A_{st+1},\ \ldots,\ A_{q-1},\ A_q]$ where A_i is the coefficient matrix of z^i in $A(z)$.

DIFFPOW – **Parameters** – `A` (*matrix*) Represents a rational matrix $A(z)$.
– `r` (*integer*) Row dimension of $A(z)$.
– `m` (*integer*) Column dimension of $A(z)$.

– **Purpose** – Computes the greatest difference between the degree of numerator and degree of denominator for each element of $A(s)$.

4.1.3 Forward Solution – MAPLE code

`FORWARD(A,B,q)` procedure

```
FORWARD:=proc(A,B,q)
  local r,m,Aseries,Bseries,IA,qr,IAS,i,j,Hseries,termA,v,w,\
    In,C,termB,z,INHone,t,INHtwo,g,k,N,Y,Z,YY,Ma,U,X,UU,Mb;
  r:=rowdim(A); m:=coldim(B);
  Aseries:=MATEXP(A,q,0,r,r);
  Bseries:=MATEXP(B,q,0,r,m);
  IA:=inverse(A);
  qr:=DIFFPOW(IA,r,r);
  IAS:=map(series,IA,s=infinity,q+qr+1);
  for i from 1 to r do
    for j from 1 to r do
      if type(op(nops(IAS[i,j]),IAS[i,j]),function)=true then
        IAS[i,j]:=sum(op(k,IAS[i,j]),k=1..nops(IAS[i,j])-1)
      fi
    od
  od;
  Hseries:=MATEXP(IAS,qr,-q,r,r);
  termA:=matrix(q*r,q*r,0);
  for v from 1 to q do
    for w from 1 to q do
      if v>=w then
        termA:=copyinto(Aseries[v-w+1],termA,(v-1)*r+1,(w-1)*r+1)
      fi
    od
  od;
  In:=Bseries[1];
  for C from 2 to q+1 do
    In:=augment(In,Bseries[C])
  od;
  termB:=matrix((q+qr+1)*r,(2*q+qr+1)*m,0);
  for z from 1 to q+qr+1 do
    termB:=copyinto(In,termB,(z-1)*r+1,(z-1)*m+1)
  od;
  INHone:=Hseries[1];
  for t from 2 to q do
```

```
    INHone:=augment(Hseries[t],INHone)
  od;
  INHtwo:=Hseries[1];
  for g from 2 to q+qr+1 do
    INHtwo:=augment(INHtwo,Hseries[g])
  od;
  for N from 1 to q do
    Y:=matrix(r,1);
    for Z from 1 to r do
      Y[Z,1]:=y.kmin.N.Z
    od;
    if N=1 then
      YY:=op(Y)
    else
      YY:=stack(YY,Y)
    fi
  od;
  for Ma from q by -1 to 0 do
    U:=matrix(m,1);
    for X from 1 to m do
      U[X,1]:=u.kmin.Ma.X
    od;
    if Ma=q then
      UU:=op(U)
    else
      UU:=stack(UU,U)
    fi
  od;
  for Mb from 1 to q+qr do
    U:=matrix(m,1);
    for X from 1 to m do
      U[X,1]:=u.kplus.Mb.X
    od;
    UU:=stack(UU,U)
  od;
  evalm(-1 * INHone &* termA &* YY + INHtwo &* termB &* UU)
end;
```

MATEXP(A,q,st,n,m) procedure

```
MATEXP:=proc(A,q,st,n,m)
  local Aexp,i,j,k,L,l;
  Aexp:=map(expand,A);
  for i from q by -1 to st do
    A.i:=matrix(n,m,0);
    for j from 1 to n do
      for k from 1 to m do
```

```
        A.i[j,k]:=coeff(Aexp[j,k],s,i)
      od
    od
  od;
  L:=[];
  for l from st to q do
    L:=[op(L),op(A.l)]
  od;
  L
end;
```

DIFFPOW(A,r,m) procedure

```
DIFFPOW:=proc(A,r,m)
  local Anorm,diff,i,j,F;
  Anorm:=map(normal,A);
  diff:=0;
  for i from 1 to r do
    for j from 1 to m do
      F:=degree(numer(Anorm[i,j]),s)-degree(denom(Anorm[i,j]),s);
      if F>diff then
        diff:=F
      fi
    od
  od;
  diff
end;
```

4.2 Backward Solution

Here we implement the alternative backward solution of the ARMA–Representation (1.1) as given in (3.8).

4.2.1 Backward Solution – Pseudocode

INPUT

– The matrices $A(z)$ and $B(z)$

– The integer $q \stackrel{\text{def}}{=}$ highest degree element in $A(z)$ or $B(z)$

PROCEDURE

$Aseries \leftarrow$ the matrix expansion $\{A_0,\ A_1,\ \ldots,\ A_q\}$

$Bseries \leftarrow$ the matrix expansion $\{B_0,\ B_1,\ \ldots,\ B_q\}$

$IA \leftarrow$ the inverse of A

$IAS \leftarrow$ the matrix IA with each term expanded about $z = 0$

$Vseries \leftarrow$ the matrix expansion $\{V_{-l},\ V_{-l+1},\ \ldots,\ V_q\}$

$$termA \leftarrow \text{the block matrix } \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix}$$

$$termB \leftarrow \text{the block matrix } \begin{pmatrix} B_q & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_0 \end{pmatrix}$$

$$INVone \leftarrow \text{the block matrix } [V_q\ V_{q-1}\ \cdots\ V_1]$$

$$INVtwo \leftarrow \text{the block matrix } [V_0\ V_{-1}\ \cdots\ V_{-l}]$$

$$YY \leftarrow \text{the vector } \begin{pmatrix} y(k+q) \\ \vdots \\ y(k+1) \end{pmatrix}$$

$$UU \leftarrow \text{the vector } \begin{pmatrix} u(k+q) \\ \vdots \\ u(k) \\ \hline u(k-1) \\ \vdots \\ u(k-l) \end{pmatrix}$$

OUTPUT

– The vector $[INVone \cdot termA \cdot YY] + [INVtwo \cdot termB \cdot UU]$

4.2.2 Backward Solution – Procedures

For the implementation of (3.8) in MAPLE two procedures are presented; firstly the main calling procedure `BACK` and secondly the sub–procedures `MATEXP`. The sub–procedure `MATEXP` is given in section 4.1.2 .

`BACK` – **Parameters**
- `A,B` (*matrices*) Represent polynomial matrices $A(z)$ and $B(z)$.
- `q` (*integer*) The highest degree element in $A(z)$ and $B(z)$.

– **Purpose**
- Computes the backward solution $y(k)$ of the ARMA–Representation $A(\sigma)y(k) = B(\sigma)u(k)$ in terms of the future q vectors $y(k+1),\ \cdots,\ y(k+q)$ and the input sequence $u(k)$.

`MATEXP` – As in 4.1.2 above

4.2.3 Backward Solution – MAPLE code

BACK(A,B,q) procedure

```
BACK:=proc(A,B,q)
  local r,m,Aseries,Bseries,IA,IAS,l,aa,bb,Vseries,termA,v,w,\
    In,C,termB,z,INVone,t,INVtwo,g,N,Y,Z,YY,Ma,U,X,UU,Mb;
  r:=rowdim(A);
  m:=coldim(B);
  Aseries:=MATEXP(A,q,0,r,r);
  Bseries:=MATEXP(B,q,0,r,m);
  IA:=inverse(A);
  IAS:=map(series,IA,s=0,6);
  IAS:=map(convert,IAS,polynom);
  l:=ldegree(IAS[1,1]);
  for aa from 1 to r do
    for bb from 1 to r do
      l:=min(l,ldegree(IAS[aa,bb]))
    od
  od;
  l:=-l;
  if (l+q+1)>6 then
    IAS:=map(series,IA,s=0,l+q+1)
  fi;
  IAS:=map(convert,IAS,polynom);
  Vseries:=MATEXP(IAS,q,-l,r,r);
  termA:=matrix(q*r,q*r,0);
  for v from 1 to q do
    for w from 1 to q do
      if v>=w then
        termA:=copyinto(Aseries[v-w+1],termA,(v-1)*r+1,(w-1)*r+1)
      fi
    od
  od;
  In:=Bseries[q+1];
  for C from q by -1 to 1 do
    In:=augment(In,Bseries[C])
  od;
  termB:=matrix((l+1)*r,(l+q+1)*m,0);
  for z from 1 to l+1 do
    termB:=copyinto(In,termB,(z-1)*r+1,(z-1)*m+1)
  od;
  INVone:=Vseries[l+q+1];
  for t from l+q by -1 to l+2 do
    INVone:=augment(INVone,Vseries[t])
  od;
  INVtwo:=Vseries[1];
```

```
  for g from 2 to l+1 do
    INVtwo:=augment(Vseries[g],INVtwo)
  od;
  for N from q by -1 to 1 do
    Y:=matrix(r,1);
    for Z from 1 to r do
      Y[Z,1]:=y.kplus.N.Z
    od;
    if N=q then
      YY:=op(Y)
    else
      YY:=stack(YY,Y)
    fi
  od;
  for Ma from q by -1 to 0 do
    U:=matrix(m,1);
    for X from 1 to m do
      U[X,1]:=u.kplus.Ma.X
    od;
    if Ma=q then
      UU:=op(U)
    else
      UU:=stack(UU,U)
    fi
  od;
  for Mb from 1 to l do
    U:=matrix(m,1);
    for X from 1 to m do
      U[X,1]:=u.kmin.Mb.X
    od;
    UU:=stack(UU,U)
  od;
  evalm(INVone &* termA &* YY + INVtwo &* termB &* UU)
end;
```

4.3 Symmetric Solution

Here we implement the symmetric solution of the ARMA–Representation (1.1) as given in (3.5). The forward–symmetric and backward–symmetric solutions (3.9) and (3.10) can be similarly implemented.

4.3.1 Symmetric Solution – Pseudocode

INPUT

– The matrices $A(z)$ and $B(z)$

– The integer $q \stackrel{\text{def}}{=}$ highest degree element in $A(z)$ or $B(z)$

– The integer $N \stackrel{\text{def}}{=}$ end of range of solution considered

– The integer $k \stackrel{\text{def}}{=}$ the required solution index

PROCEDURE

$Aseries \leftarrow$ the matrix expansion $\{A_0,\ A_1,\ \ldots,\ A_q\}$

$Bseries \leftarrow$ the matrix expansion $\{B_0,\ B_1,\ \ldots,\ B_q\}$

$IA \leftarrow$ the inverse of A

$IAS \leftarrow$ the matrix IA with each term expanded about $z = \infty$

$Hseries \leftarrow$ the matrix expansion $\{H_{-k-q},\ H_{-k-q+1},\ \ldots,\ H_{\hat{q}_r}\}$

$$termAUP \leftarrow \text{the block matrix } \begin{pmatrix} A_q & \cdots & A_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_q \end{pmatrix}$$

$INHone \leftarrow$ the block matrix $[H_{-k-1}\ H_{-2}\ \cdots\ H_{-k-q}]$

$$SS \leftarrow \text{the vector } \begin{pmatrix} y(q-1) \\ \vdots \\ y(0) \end{pmatrix}$$

if $(N-k-q+1) \leq \hat{q}_r$

$$termALO \leftarrow \text{the block matrix } \begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix}$$

$INHtwo \leftarrow$ the block matrix $[H_{N-k}\ H_{N-k-1}\ \cdots\ H_{N-k-q+1}]$

$$PP \leftarrow \text{the vector } \begin{pmatrix} y(N) \\ \vdots \\ y(N-q+1) \end{pmatrix}$$

end if

$INHthree \leftarrow$ the block matrix $[H_{N-k-q}\ H_{N-k-q-1}\ \cdots\ H_{-k}]$

$$termB \leftarrow \text{the block matrix } \begin{pmatrix} B_q & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_0 \end{pmatrix}$$

$$UU \leftarrow \text{the vector } \begin{pmatrix} u(N) \\ \vdots \\ u(0) \end{pmatrix}$$

OUTPUT

if $(N-k-q+1) > \hat{q}_r$

– The vector $[INHone \cdot termAUP \cdot YY] + [INHthree \cdot termB \cdot UU]$

else

– The vector $[INHone \cdot termAUP \cdot YY] + [INHtwo \cdot termALO \cdot WW] +$
$+ [INHthree \cdot termB \cdot UU]$

end if

4.3.2 Symmetric Solution – Procedures

For the implementation of (3.5) in MAPLE three procedures are presented; firstly the main calling procedure `FORWARD` and secondly the two sub–procedures `MATEXP` and `DIFFPOW`. The sub–procedures `MATEXP` and `DIFFPOW` are as given in section 4.1.2 .

`SYMMETRIC` – **Parameters**
- `A,B` (*matrices*) Represent polynomial matrices $A(z)$ and $B(z)$.
- `q` (*integer*) The highest degree element in $A(z)$ and $B(z)$.
- `N` (*integer*) The end of range of solution considered.
- `k` (*integer*) The required solution index.

– **Purpose**
- Computes the forward solution $y(k)$ of the ARMA–Representation $A(\sigma)y(k) = B(\sigma)u(k)$ in terms of the previous q vectors $y(k-1)$, $\cdots$, $y(k-q)$ and the input sequence $u(k)$.

`MATEXP` – As in 4.1.2 above.

`DIFFPOW` – As in 4.1.2 above.

4.3.3 Symmetric Solution – MAPLE code

`SYMMETRIC(A,B,q,N,k)` procedure

```
SYMMETRIC:=proc(A,B,q,N,k)
  local r,m,Aseries,Bseries,IA,qr,IAS,i,j,kk,Hseries,termAUP,v,w,\
    INHone,t,SS,Y,Z,YY,termALO,INHtwo,TT,PP,W,WW,INHthree,stval,BB,\
    In,C,termB,z,KK,U,X,UU;
  r:=rowdim(A);
  m:=coldim(B);
  Aseries:=MATEXP(A,q,0,r,r);
  Bseries:=MATEXP(B,q,0,r,m);
  IA:=inverse(A);
  qr:=DIFFPOW(IA,r,r);
  IAS:=map(series,IA,s=infinity,qr+k+q+1);
  for i from 1 to r do
    for j from 1 to r do
      if type(op(nops(IAS[i,j]),IAS[i,j]),function)=true then
```

```
      IAS[i,j]:=sum(op(kk,IAS[i,j]),kk=1..nops(IAS[i,j])-1)
    fi
  od
od;
Hseries:=MATEXP(IAS,qr,-k-q,r,r);
termAUP:=matrix(q*r,q*r,0);
for v from 1 to q do
  for w from 1 to q do
    if v<=w then
      termAUP:=copyinto(Aseries[v-w+q+1],termAUP,(v-1)*r+1,(w-1)*r+1)
    fi
  od
od;
INHone:=Hseries[1];
for t from 2 to q do
  INHone:=augment(Hseries[t],INHone)
od;
for SS from 1 to q do
  Y:=matrix(r,1);
  for Z from 1 to r do
    Y[Z,1]:=y.(q-SS).Z
  od;
  if SS=1 then
    YY:=op(Y)
  else
    YY:=stack(YY,Y)
  fi
od;
if (N-k-q+1)<=qr then
  termALO:=matrix(q*r,q*r,0);
  for v from 1 to q do
    for w from 1 to q do
      if v>=w then
        termALO:=copyinto(Aseries[v-w+1],termALO,(v-1)*r+1,(w-1)*r+1)
      fi
    od
  od;
  INHtwo:=Hseries[N+2];
  for TT from N+3 to min(N+q+1,k+qr+q+1) do
    INHtwo:=augment(Hseries[TT],INHtwo)
  od;
  INHtwo:=copyinto(INHtwo,matrix(r,q*r,0),1,q*r-coldim(INHtwo)+1);
  for PP from 0 to q-1 do
    W:=matrix(r,1);
    for Z from 1 to r do
      W[Z,1]:=y.(N-PP).Z
```

```
      od;
      if PP=0 then
        WW:=op(W)
      else
        WW:=stack(WW,W)
      fi
    od
  fi;
  INHthree:=Hseries[q+1];
  stval:=min(k+qr+1,N-q+1);
  for BB from 2 to stval do
    INHthree:=augment(Hseries[q+BB],INHthree)
  od;
  In:=Bseries[1];
  for C from 2 to q+1 do
    In:=augment(Bseries[C],In)
  od;
  termB:=matrix((stval)*r,(stval+q)*m,0);
  for z from 1 to stval do
    termB:=copyinto(In,termB,(z-1)*r+1,(z-1)*m+1)
  od;
  for KK from 0 to stval+q-1 do
    U:=matrix(m,1);
    for X from 1 to m do
      U[X,1]:=u.(KK).X
    od;
    if KK=0 then
      UU:=op(U)
    else
      UU:=stack(U,UU)
    fi
  od;
  if (N-k-q+1)>qr then
    evalm(INHone &* termAUP &* YY + INHthree &* termB &* UU)
  else
    evalm(INHone &* termAUP &* YY + INHtwo &* termALO &* WW \
      + INHthree &* termB &* UU)
  fi
end;
```

4.4 Forward Compatibility Condition

Here we implement the forward compatibility condition of the ARMA–Representation (1.1) as given in (3.2).

4.4.1 Forward Compatibility Condition – Pseudocode

INPUT

– The matrices $A(z)$ and $B(z)$

– The integer $q \stackrel{\text{def}}{=}$ highest degree element in $A(z)$ or $B(z)$

– The **optional** variable L where the solution is stored

PROCEDURE

$Aseries \leftarrow$ the matrix expansion $\{A_0,\ A_1,\ \ldots,\ A_q\}$

$Bseries \leftarrow$ the matrix expansion $\{B_0,\ B_1,\ \ldots,\ B_q\}$

$IA \leftarrow$ the inverse of A

$IAS \leftarrow$ the matrix IA with each term expanded about $z = \infty$

$Hseries \leftarrow$ the matrix expansion $\{H_{-q+1},\ H_{-q+2},\ \ldots,\ H_{\hat{q}_r}\}$

$$termA \leftarrow \text{the block matrix } \begin{pmatrix} A_0 & \cdots & A_{q-1} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_0 \end{pmatrix}$$

$$termB \leftarrow \text{the block matrix } \begin{pmatrix} B_0 & \cdots & B_q & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_0 & \cdots & B_q \end{pmatrix}$$

$$termHL \leftarrow \text{the block matrix } \begin{pmatrix} H_0 & \cdots & H_{q-1} \\ \vdots & \ddots & \vdots \\ H_{-q+1} & \cdots & H_0 \end{pmatrix}$$

$$termHL \leftarrow \text{the block matrix } \begin{pmatrix} H_0 & \cdots & H_{\hat{q}_r} & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ H_{-q+1} & \cdots & H_{\hat{q}_r-q+1} & \cdots & H_{\hat{q}_r} \end{pmatrix}$$

$$YY \leftarrow \text{the vector } \begin{pmatrix} y(0) \\ \vdots \\ y(q-1) \end{pmatrix}$$

$$UU \leftarrow \text{the vector } \begin{pmatrix} u(0) \\ \vdots \\ u(2q+\hat{q}_r-1) \end{pmatrix}$$

if number of arguments > 3 then

if solution to (3.2) exists

$$L \leftarrow \text{the admissible solution to (3.2) in the form } \left(\begin{array}{c} y(0) \\ \vdots \\ y(q-1) \\ \hline -u(0) \\ \vdots \\ -u(2q+\hat{q}_r-1) \end{array}\right)$$

else if no solution to (3.2) exists

$L \leftarrow$ 'No solution'

end if

end if

OUTPUT

– The matrix equation $[termA \cdot termHL \cdot termA] \cdot YY =$

$= [termA \cdot termHR \cdot termB] \cdot UU$

4.4.2 Forward Compatibility Condition – Procedures

For the implementation of (3.2) in MAPLE three procedures are presented; firstly the main calling procedure `FORWARD` and secondly the two sub–procedures `MATEXP` and `DIFFPOW`. The sub–procedures `MATEXP` and `DIFFPOW` are as given in section 4.1.2 .

`FORADMISS`

– **Parameters**

- `A,B` (*matrices*) Represent polynomial matrices $A(z)$ and $B(z)$.
- `q` (*integer*) The highest degree element in $A(z)$ and $B(z)$.
- `L` (*integer*) OPTIONAL. If included the solution of the forward compatibility condition (3.2) is stored in L.

– **Purpose**

- Computes the forward compatibility condition of the ARMA–Representation $A(\sigma)y(k) = B(\sigma)u(k)$. Computes the solution of this if the optional parameter L is included.

`MATEXP` – As in 4.1.2 above.

`DIFFPOW` – As in 4.1.2 above.

4.4.3 Forward Compatibility Condition – MAPLE code

FORWADMISS(A,B,q,L) procedure

```
FORADMISS:=proc(A,B,q,L)
  local r,m,Aseries,Bseries,IA,qr,IAS,i,j,Hseries,termA,v,w,Q,\
    In,C,termB,z,termHL,ih,jh,pm,InHR,sm,termHR,N,Y,Z,YY,M,U,X,UU,BB,CC;
  r:=rowdim(A);
  m:=coldim(B);
  Aseries:=MATEXP(A,q,0,r,r);
  Bseries:=MATEXP(B,q,0,r,m);
  IA:=inverse(A);
  qr:=DIFFPOW(IA,r,r);
  Q:=max(q-1,qr);
  IAS:=map(series,IA,s=infinity,q+Q);
  for i from 1 to r do
    for j from 1 to r do
      if type(op(nops(IAS[i,j]),IAS[i,j]),function)=true then
        IAS[i,j]:=sum(op(k,IAS[i,j]),k=1..nops(IAS[i,j])-1)
      fi
    od
  od;
  Hseries:=MATEXP(IAS,Q,-q+1,r,r);
  termA:=matrix(q*r,q*r,0);
  for v from 1 to q do
    for w from 1 to q do
      if v<=w then
        termA:=copyinto(Aseries[w-v+1],termA,(v-1)*r+1,(w-1)*r+1)
      fi
    od
  od;
  In:=Bseries[1];
  for C from 2 to q+1 do
    In:=augment(In,Bseries[C])
  od;
  termB:=matrix((q+qr)*r,(2*q+qr)*m,0);
  for z from 1 to q+qr do
    termB:=copyinto(In,termB,(z-1)*r+1,(z-1)*m+1)
  od;
  termHL:=matrix(q*r,q*r,0);
  for ih from 1 to q do
    for jh from 1 to q do
      termHL:=copyinto(Hseries[jh+q-ih],termHL,(ih-1)*r+1,(jh-1)*r+1)
    od
  od;
  for pm from 1 to q do
    InHR:=Hseries[pm];
```

```
    for sm from pm+1 to q+qr do
      InHR:=augment(InHR,Hseries[sm])
    od;
    InHR:=copyinto(InHR,matrix(r,(q+qr)*r,0),1,1);
    if pm=1 then
      termHR:=op(InHR)
    else
      termHR:=stack(InHR,termHR)
    fi
  od;
  for N from 0 to q-1 do
    Y:=matrix(r,1);
    for Z from 1 to r do
      Y[Z,1]:=y.kplus.N.Z
    od;
    if N=0 then
      YY:=op(Y)
    else
      YY:=stack(YY,Y)
    fi
  od;
  for M from 0 to 2*q+qr-1 do
    U:=matrix(m,1);
    for X from 1 to m do
      U[X,1]:=u.kplus.M.X
    od;
    if M=0 then
      UU:=op(U)
    else
      UU:=stack(UU,U)
    fi
  od;
  if nargs>3 then
    BB:=augment(evalm(termA &* termHL &* termA),\
      evalm(termA &* termHR &* termB));
    CC:=linsolve(BB,matrix(q*r,1,0));
    if CC=NULL then
      L:=print(`No solution`)
    else
      L:=CC
    fi
  fi;
  print(evalm(termA &* termHL &* termA),op(YY),`=`,\
    evalm(termA &* termHR &* termB),op(UU))
end;
```

4.5 Backward Compatibility Condition

Here we implement the backward compatibility condition of the ARMA–Representation (1.1) as given in (3.4).

4.5.1 Backward Compatibility Condition – Pseudocode

INPUT

– The matrices $A(z)$ and $B(z)$

– The integer $q \stackrel{\text{def}}{=}$ highest degree element in $A(z)$ or $B(z)$

– The **optional** variable L where the solution is stored

PROCEDURE

$Aseries \leftarrow$ the matrix expansion $\{A_0,\ A_1,\ \ldots,\ A_q\}$

$Bseries \leftarrow$ the matrix expansion $\{B_0,\ B_1,\ \ldots,\ B_q\}$

$IA \leftarrow$ the inverse of A

$IAS \leftarrow$ the matrix IA with each term expanded about $z = 0$

$-l \leftarrow$ lowest z degree out of every term element of IAS.

if $l < 1$ then

RETURN 'No restrictions apply'

end if

$Vseries \leftarrow$ the matrix expansion $\{V_{-l},\ V_{-l+1},\ \ldots,\ V_{-1}\}$

$$termA \leftarrow \text{the block matrix } \begin{pmatrix} A_q & \cdots & A_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_q \end{pmatrix}$$

$$termB \leftarrow \text{the block matrix } \begin{pmatrix} B_q & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_0 \end{pmatrix}$$

$$INVone \leftarrow \text{the block matrix } \begin{pmatrix} V_{-q} & \cdots & V_{-2q+1} \\ \vdots & \ddots & \vdots \\ V_{-1} & \cdots & V_{-q} \end{pmatrix}$$

$$termHL \leftarrow \text{the block matrix } \begin{pmatrix} V_{-q} & \cdots & V_{-l} & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ V_{-1} & \cdots & V_{-l+q-1} & \cdots & V_{-l} \end{pmatrix}$$

$$YY \leftarrow \text{the vector } \begin{pmatrix} y(N) \\ \vdots \\ y(N-q+1) \end{pmatrix}$$

$UU \leftarrow$ the vector $\begin{pmatrix} u(N) \\ \vdots \\ u(N-q-l+1) \end{pmatrix}$

if number of arguments > 3 then

if solution to (3.4) exists

$L \leftarrow$ the admissible solution to (3.4) in the form $\begin{pmatrix} y(N) \\ \vdots \\ y(N-q+1) \\ \hline -u(N) \\ \vdots \\ -u(N-q-l+1) \end{pmatrix}$

else if no solution to (3.4) exists

$L \leftarrow$ 'No solution'

end if

end if

OUTPUT

– The matrix equation $[termA \cdot INVone \cdot termA] \cdot YY =$

$= [termA \cdot INVtwo \cdot termB] \cdot UU$

4.5.2 Backward Compatibility Condition – Procedures

For the implementation of (3.4) in MAPLE two procedures are presented; firstly the main calling procedure `FORWARD` and secondly the sub–procedure `MATEXP`. The sub–procedure `MATEXP` is as given in section 4.1.2 .

`BACKADMISS` – **Parameters**

- `A,B` (*matrices*) Represent polynomial matrices $A(z)$ and $B(z)$.
- `q` (*integer*) The highest degree element in $A(z)$ and $B(z)$.
- `L` (*integer*) OPTIONAL. If included the solution of the backward compatibility condition (3.4) is stored in L.

– **Purpose**

- Computes the backward compatibility condition of the ARMA–Representation $A(\sigma)y(k) = B(\sigma)u(k)$. Computes the solution of this if the optional parameter L is included.

`MATEXP` – As in 4.1.2 above.

`DIFFPOW` – As in 4.1.2 above.

4.5.3 Forward Compatibility Condition – MAPLE code

FORWADMISS(A,B,q,L) procedure

```
BACKADMISS:=proc(A,B,q,L)
  local r,m,Aseries,Bseries,IA,IAS,l,aa,bb,Q,Vseries,termA,v,w,\
    In,C,termB,z,INVone,tt,dd,gg,Vtwo,INVtwo,ww,TT,Y,Z,YY,M,U,X,UU,BB,CC;
  r:=rowdim(A);
  m:=coldim(B);
  Aseries:=MATEXP(A,q,0,r,r);
  Bseries:=MATEXP(B,q,0,r,m);
  IA:=inverse(A);
  IAS:=map(series,IA,s=0,6);
  IAS:=map(convert,IAS,polynom);
  l:=ldegree(IAS[1,1]);
  for aa from 1 to r do
    for bb from 1 to r do
      l:=min(l,ldegree(IAS[aa,bb]))
    od
  od;
  l:=-l;
  if l<1 then
    RETURN(‘No restrictions apply‘)
  elif l>6 then
    IAS:=map(series,IA,s=0,l);
  fi;
  IAS:=map(convert,IAS,polynom);
  Q:=-min(-l,-2*q+1);
  Vseries:=MATEXP(IAS,-1,-Q,r,r);
  termA:=matrix(q*r,q*r,0);
  for v from 1 to q do
    for w from 1 to q do
      if v<=w then
        termA:=copyinto(Aseries[-w+q+v+1],termA,(v-1)*r+1,(w-1)*r+1)
      fi
    od
  od;
  In:=Bseries[q+1];
  for C from q by -1 to 1 do
    In:=augment(In,Bseries[C])
  od;
  termB:=matrix(l*r,(l+q)*m,0);
  for z from 1 to l do
    termB:=copyinto(In,termB,(z-1)*r+1,(z-1)*m+1)
  od;
  INVone:=matrix(q*r,q*r,0);
  for tt from 1 to q do
```

```
  for dd from 1 to q do
    INVone:=copyinto(Vseries[Q-q+tt-dd+1],INVone,(tt-1)*r+1,(dd-1)*r+1)
  od
od;
for gg from 1 to q do
  Vtwo:=Vseries[Q+1-gg];
  for ww from gg+1 to l do
    Vtwo:=augment(Vtwo,Vseries[Q+1-ww])
  od;
  Vtwo:=copyinto(Vtwo,matrix(r,l*r,0),1,1);
  if gg=1 then
    INVtwo:=op(Vtwo)
  else
    INVtwo:=stack(Vtwo,INVtwo)
  fi
od;
for TT from 0 to q-1 do
  Y:=matrix(r,1);
  for Z from 1 to r do
    Y[Z,1]:=y.Nmin.TT.Z
  od;
  if TT=0 then
    YY:=op(Y)
  else
    YY:=stack(YY,Y)
  fi
od;
for M from 0 to q+l-1 do
  U:=matrix(m,1);
  for X from 1 to m do
    U[X,1]:=u.Nmin.M.X
  od;
  if M=0 then
    UU:=op(U)
  else
    UU:=stack(UU,U)
  fi
od;
if nargs>3 then
  BB:=augment(evalm(termA &* INVone &* termA),\
    evalm(termA &* INVtwo &* termB));
  CC:=linsolve(BB,matrix(q*r,1,0));
  if CC=NULL then
    L:=print(‘No solution‘)
  else
    L:=CC
```

```
    fi
  fi;
  print(evalm(termA &* INVone &* termA),op(YY),'=',\
    evalm(termA &* INVtwo &* termB),op(UU))
end;
```

4.6 Symmetric Boundary Mapping Equation

Here we implement the boundary mapping equation of the ARMA–Representation (1.1) as given in (3.6).

4.6.1 Boundary Mapping Equation – Pseudocode

INPUT

– The matrices $A(z)$ and $B(z)$

– The integer $q \stackrel{\text{def}}{=}$ highest degree element in $A(z)$ or $B(z)$

– The end of interval range N

– The **optional** variable L where the solution is stored

PROCEDURE

$Aseries \leftarrow$ the matrix expansion $\{A_0,\ A_1,\ \ldots,\ A_q\}$

$Bseries \leftarrow$ the matrix expansion $\{B_0,\ B_1,\ \ldots,\ B_q\}$

$IA \leftarrow$ the inverse of A

$IAS \leftarrow$ the matrix IA with each term expanded about $z = \infty$

$Hseries \leftarrow$ the matrix expansion $\{H_{-N},\ H_{-N+!},\ \ldots,\ H_{\hat{q}_r}\}$

$$Waa \leftarrow \text{the block matrix } \begin{pmatrix} H_{-q} & \cdots & H_{-2q+1} \\ \vdots & \ddots & \vdots \\ H_{-1} & \cdots & H_{-q} \end{pmatrix}$$

$$Wab \leftarrow \text{the block matrix } \begin{pmatrix} H_{-N+q-1} & \cdots & H_{-N} \\ \vdots & \ddots & \vdots \\ H_{-N+2q-2} & \cdots & H_{-N+q-1} \end{pmatrix}$$

if $(N - 3q + 2) > \hat{q}_r$ then

$Wba \leftarrow$ the block matrix $0_{(qr \times qr)}$

else

$$Wba \leftarrow \text{the block matrix } \begin{pmatrix} H_{N-2q+1} & \cdots & H_{N-3q+2} \\ \vdots & \ddots & \vdots \\ H_{N-q} & \cdots & H_{N-2q+1} \end{pmatrix}$$

end if

$Wbb \leftarrow$ the block matrix $\begin{pmatrix} H_0 & \cdots & H_{-q+1} \\ \vdots & \ddots & \vdots \\ H_{q-1} & \cdots & H_0 \end{pmatrix}$

$XAbar \leftarrow$ the block matrix $\begin{pmatrix} A_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ A_{q-1} & \cdots & A_0 \end{pmatrix}$

$XA \leftarrow$ the block matrix $\begin{pmatrix} A_q & \cdots & A_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_q \end{pmatrix}$

$Zone \leftarrow$ the block matrix $\begin{pmatrix} H_{-q} & \cdots & H_{-N} \\ \vdots & \ddots & \vdots \\ H_{-1} & \cdots & H_{-N+q-1} \end{pmatrix}$

$Ztwo \leftarrow$ the block matrix $\begin{pmatrix} H_{N-2q+1} & \cdots & H_{-q+1} \\ \vdots & \ddots & \vdots \\ H_{N-q} & \cdots & H_0 \end{pmatrix}$

$termB \leftarrow$ the block matrix $\begin{pmatrix} B_q & \cdots & B_0 & \cdots & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & \cdots & B_q & \cdots & B_0 \end{pmatrix}$

$WW \leftarrow$ the vector $\begin{pmatrix} y(N) \\ \vdots \\ y(N-q+1) \end{pmatrix}$

$YY \leftarrow$ the vector $\begin{pmatrix} y(q-1) \\ \vdots \\ y(0) \end{pmatrix}$

$UU \leftarrow$ the vector $\begin{pmatrix} u(N) \\ \vdots \\ u(0) \end{pmatrix}$

if number of arguments > 4 then

if solution to (3.6) exists

$L \leftarrow$ the admissible solution to (3.6) in the form $\left(\begin{array}{c} y(N) \\ \vdots \\ \hline y(N-q+1) \\ y(q-1) \\ \vdots \\ \hline y(0) \\ -u(N) \\ \vdots \\ -u(0) \end{array}\right)$

else if no solution to (3.2) exists

$L \leftarrow$ 'No solution'

end if

end if

OUTPUT

– The matrix equation

$$\left[\left(\begin{array}{c|c} Waa \cdot XA & Wab \cdot XAbar \\ \hline Wba \cdot XA & Wbb \cdot XAbar \end{array}\right)\right]\left(\begin{array}{c} WW \\ \hline YY \end{array}\right) = \left[\left(\begin{array}{c} Zone \\ \hline Ztwo \end{array}\right) termB\right] UU$$

4.6.2 Boundary Mapping Equation – Procedures

For the implementation of (3.6) in MAPLE three procedures are presented; firstly the main calling procedure `FORWARD` and secondly the two sub–procedures `MATEXP` and `DIFFPOW`. The sub–procedures `MATEXP` and `DIFFPOW` are as given in section 4.1.2 .

`SYMADMISS` – **Parameters**

- `A,B` (*matrices*) Represent polynomial matrices $A(z)$ and $B(z)$.
- `q` (*integer*) The highest degree element in $A(z)$ and $B(z)$.
- `N` (*integer*) The range of solution $[0,\ N]$.
- `L` (*integer*) OPTIONAL. If included the solution of the boundary mapping equation (3.6) is stored in L.

– **Purpose**

- Computes the boundary mapping equation of the ARMA–Representation $A(\sigma)y(k) = B(\sigma)u(k)$. Computes the solution of this if the optional parameter L is included.

`MATEXP` – As in 4.1.2 above.

`DIFFPOW` – As in 4.1.2 above.

4.6.3 Forward Compatibility Condition – MAPLE code

SYMADMISS(A,B,q,N,L) procedure

```
SYMADMISS:=proc(A,B,q,N,L)
  local r,m,Aseries,Bseries,IA,qr,IAS,i,j,kk,Hseries,Waa,a,b,Wab,\
    c,d,Wba,e,f,Wbb,g,h,XAbar,j,k,XA,t,u,Zone,v,w,Ztwo,x,z,In,C,\
    termB,DD,EE,W,Z,WW,FF,Y,T,YY,GG,U,X,UU,BB,CC,KK,U,X,UU;
  r:=rowdim(A);
  m:=coldim(B);
  Aseries:=MATEXP(A,q,0,r,r);
  Bseries:=MATEXP(B,q,0,r,m);
  IA:=inverse(A);
  qr:=DIFFPOW(IA,r,r);
  IAS:=map(series,IA,s=infinity,qr+N+1);
  for i from 1 to r do
    for j from 1 to r do
      if type(op(nops(IAS[i,j]),IAS[i,j]),function)=true then
        IAS[i,j]:=sum(op(kk,IAS[i,j]),kk=1..nops(IAS[i,j])-1)
      fi
    od
  od;
  Hseries:=MATEXP(IAS,qr,-N,r,r);
  Waa:=matrix(q*r,q*r,0);
  for a from 1 to q do
    for b from 1 to q do
      Waa:=copyinto(Hseries[N+1-q+a-b],Waa,(a-1)*r+1,(b-1)*r+1)
    od
  od;
  Wab:=matrix(q*r,q*r,0);
  for c from 1 to q do
    for d from 1 to q do
      Wab:=copyinto(Hseries[q+c-d],Wab,(c-1)*r+1,(d-1)*r+1)
    od
  od;
  Wba:=matrix(q*r,q*r,0);
  if N-3*q+2<=qr then
    for e from 1to q do
      for f from 1 to q do
        if (N-2*q+e-f+2)<=qr+1 then
          Wba:=copyinto(Hseries[2*N-2*q+e-f+2],Wba,(e-1)*r+1,(f-1)*r+1)
        fi
      od
    od
  fi;
  Wbb:=matrix(q*r,q*r,0);
  for g from 1 to q do
```

```
  for h from 1 to q do
    if g-h<=qr then
      Wbb:=copyinto(Hseries[N+1+g-h],Wbb,(g-1)*r+1,(h-1)*r+1)
    fi
  od
od;
XAbar:=matrix(q*r,q*r,0);
for j from 1 to q do
  for k from 1 to q do
    if j>=k then
      XAbar:=copyinto(Aseries[j-k+1],XAbar,(j-1)*r+1,(k-1)*r+1)
    fi
  od
od;
XA:=matrix(q*r,q*r,0);
for t from 1 to q do
  for u from 1 to q do
    if t<=u then
      XA:=copyinto(Aseries[q+1+t-u],XA,(t-1)*r+1,(u-1)*r+1)
    fi
  od
od;
Zone:=matrix(q*r,(N-q+1)*r,0);
for v from 1 to q do
  for w from 1 to N-q+1 do
    Zone:=copyinto(Hseries[N-q+1+v-w],Zone,(v-1)*r+1,(w-1)*r+1)
  od
od;
Ztwo:=matrix(q*r,(N-q+1)*r,0);
for x from 1 to q do
  for z from 1 to N-q+1 do
    if (N-2*q+x-z+2)<=qr+1 then
      Ztwo:=copyinto(Hseries[2*N-2*q+x-z+2],Ztwo,(x-1)*r+1,(z-1)*r+1)
    fi
  od
od;
In:=Bseries[q+1];
for C from q by -1 to 1 do
  In:=augment(In,Bseries[C])
od;
termB:=matrix((N-q+1)*r,(N+1)*m,0);
for DD from 1 to N-q+1 do
  termB:=copyinto(In,termB,(DD-1)*r+1,(DD-1)*m+1)
od;
for EE from 0 to q-1 do
  W:=matrix(r,1);
```

```
    for Z from 1 to r do
      W[Z,1]:=y.(N-EE).Z
    od;
    if EE=0 then
      WW:=op(W)
    else
      WW:=stack(WW,W)
    fi
  od;
  for FF from 0 to q-1 do
    Y:=matrix(r,1);
    for T from 1 to r do
      Y[T,1]:=y.FF.T
    od;
    if FF=0 then
      YY:=op(Y)
    else
      YY:=stack(Y,YY)
    fi
  od;
  for GG from 0 to N do
    U:=matrix(m,1);
    for X from 1 to m do
      U[X,1]:=u.GG.X
    od;
    if GG=0 then
      UU:=op(U)
    else
      UU:=stack(U,UU)
    fi
  od;
  if nargs>4 then
    BB:=blockmatrix(2,3,evalm(Waa &* XA),evalm(Wab &* XAbar),\
      evalm(Zone &* termB),evalm(Wba &* XA),evalm(Wbb &* XAbar),\
      evalm(Ztwo &* termB));
    CC:=linsolve(BB,matrix(2*q*r,1,0));
    if CC=NULL then
      L:=print(`No solution`)
    else
      L:=CC
    fi
  fi;
  print(blockmatrix(2,2,evalm(Waa &* XA),evalm(Wab &* XAbar),\
    evalm(Wba &* XA),evalm(Wbb &* XAbar)),stack(WW,YY),`=`,\
    evalm(stack(Zone,Ztwo) &* termB),op(UU))
end;
```

4.7 Running Procedures

The procedures as detailed above all call several in–built procedures from the `linalg` package which exists within MAPLE. These include for example the procedures `rowdim`, `matrix` and `augment`. Initially when a MAPLE session commences these procedures are *not* loaded into MAPLE's memory and need to be loaded separately via `>with(linalg,`<*function*>`):` where <*function*> denotes a procedure which exists in the `linalg` package. Alternatively the entire `linalg` package can be read into the MAPLE session via `>with(linalg):`. This is recommended in our case as several `linalg` procedures are required. Therefore we have the following

i) Read in the linear algebra package contained within MAPLE via `> with(linalg):`

ii) Read in the solution code required via `>read('`<*proc*>`')` where <*proc*> may be either `for.m, back.m` or `sym.m` for the respective forward, backward and symmetric solution procedures. The corresponding compatibility conditions similarly have the form `FORADMISS.m, BACKADMISS.m` and `SYMADMISS.m` but only include the procedures `FORADMISS, BACKADMISS` and `SYMADMISS` respectively. This is because it is envisaged that they will be run in conjunction with the corresponding solution code and therefore any sub–procedures required will already be loaded into the session.

iii) Read in any required external parameters such as the matrices `A, B`.

iv) Implement the read–in solution procedure via `>FORWARD(A,B,q), >BACK(A,B,q)` or `>SYM(A,B,q,N,k)` or the corresponding compatibility condition via `>FORADMISS(A,B,q,'L'), >BACKADMISS(A,B,q,'L')` or `>SYMADMISS(A,B,q,N,'L')`.

5 Examples

In this section we will consider the solution of the ARMA–Representation

$$\underbrace{\begin{pmatrix} \sigma^2+5\sigma+6 & \sigma+1 & 0 \\ 2\sigma-5 & 3\sigma+2 & 1 \\ 0 & -1 & 0 \end{pmatrix}}_{A(\sigma)} \underbrace{\begin{pmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \end{pmatrix}}_{y(k)} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}_{B(\sigma)} u(k) \tag{5.1}$$

using the MAPLE procedures presented in section 4. Clearly from (5.1) we have $q = 2, r = 3$ and $m = 1$.

The machine used is a SUN SPARC station10 (75MHz SuperSPARC II). The last line of the output indicates the CPU time used in the computation. This is divided into three parts

i) `bytes used` - *(integer)* Number of bytes of memory that have been requested up to that point in the execution of the session

ii) `alloc` - *(integer)* Number of bytes of memory *actually allocated* for data space during the session

iii) `time` - *(floating point number)* Total CPU time in seconds for the session

The output *variables* obtained from the procedures will be of the following form. Consider , for example, a vector $Y \in \mathbb{R}^{(q+1)r}$ which is represented below.

$$Y = \begin{pmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+q) \end{pmatrix} = \underbrace{\left(\begin{array}{c} y_1(k) \\ \vdots \\ y_r(k) \\ \hline y_1(k+1) \\ \vdots \\ y_r(k+1) \\ \hline \vdots \\ \vdots \\ \hline y_1(k+q) \\ \vdots \\ y_r(k+q) \end{array}\right)}_{\text{Defined vector } Y} = \underbrace{\left(\begin{array}{c} ykplus01 \\ \vdots \\ ykplus0r \\ \hline ykplus11 \\ \vdots \\ ykplus1r \\ \hline \vdots \\ \vdots \\ \hline ykplusq1 \\ \vdots \\ ykplusqr \end{array}\right)}_{\text{Vector } Y \text{ ouput via MAPLE}} \tag{5.2}$$

Also _ `t[i]` for integer values of i represents a free parameter in the solution.

5.1 Forward Solution – Example

Consider the ARMA–Representation as given in (5.1). Here we find its forward solution and an admissible set of initial input and output conditions for a solution to exist.

```
dalek%maple
    |\^/|     Maple V Release 3 (Loughborough University)
._|\|   |/|_. Copyright (c) 1981-1994 by Waterloo Maple Software and the
 \  MAPLE  /  University of Waterloo. All rights reserved. Maple and Maple V
 <____ ____>  are registered trademarks of Waterloo Maple Software.
      |       Type ? for help.
> with(linalg):
Warning: new definition for   norm
Warning: new definition for   trace
> read `for.m`;
> read `FORADMISS.m`;
> A:=matrix(3,3,[s^2+5*s+6,s+1,0,2*s-5,3*s+2,1,0,-1,0]);

                          [  2                        ]
                          [ s  + 5 s + 6   s + 1   0 ]
                          [                           ]
                     A := [    2 s - 5    3 s + 2  1 ]
                          [                           ]
                          [       0          -1    0 ]
```

```
> B:=matrix(3,1,[0,0,1]);
```

$$B := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

```
> FORWARD(A,B,2);
```

$$\begin{bmatrix} -5\,ykmin11 - 6\,ykmin21 - 5\,ykmin22 - 4\,ukmin21 + ukmin11 \end{bmatrix}$$

$$\begin{bmatrix} -ukmin01 \end{bmatrix}$$

$$\begin{bmatrix} -63\,ykmin11 - 90\,ykmin21 - 63\,ykmin22 - 48\,ukmin21 + 13\,ukmin11 + 3\,ukplus11 \end{bmatrix}$$

```
> FORADMISS(A,B,2,'L');
```

$$\begin{bmatrix} 0 & -4 & 0 & 0 & 1 & 0 \\ -5 & 2 & 1 & 2 & 3 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 & 1 & 0 \\ -12 & -10 & 0 & -15 & 2 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} ykplus01 \\ ykplus02 \\ ykplus03 \\ ykplus11 \\ ykplus12 \\ ykplus13 \end{bmatrix}, =, \begin{bmatrix} 4 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 6 & -1 & 0 & 0 & 0 \\ 8 & -2 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} ukplus01 \\ ukplus11 \\ ukplus21 \\ ukplus31 \\ ukplus41 \end{bmatrix}$$

```
> print(L);
```

$$\begin{bmatrix} \_t[1] \\ \\ \_t[3] \\ \\ 5\,\_t[1] - 2\,\_t[3] - 2\,\_t[2] - 3\,\_t[4] \\ \\ \_t[2] \\ \\ \_t[4] \\ \\ 12\,\_t[1] + 2\,\_t[3] + 15\,\_t[2] - 3\,\_t[5] \\ \\ \_t[3] \\ \end{bmatrix}$$

```
[                    _t[4]                    ]
[                                             ]
[                    _t[5]                    ]
[                                             ]
[                    _t[6]                    ]
[                                             ]
[                    _t[7]                    ]

bytes used=1556284, alloc=851812, time=1.97
dalek%
```

Here $L \in \mathbb{R}^{11}$ defines an admissible set of initial input and output conditions. Infact L is of the form

$$\begin{pmatrix} y_1(0) \\ y_2(0) \\ y_3(0) \\ y_1(1) \\ y_2(1) \\ y_3(1) \\ -u(0) \\ -u(1) \\ -u(2) \\ -u(3) \\ -u(4) \end{pmatrix} \tag{5.3}$$

5.2 Backward Solution – Example

Consider the ARMA–Representation as given in (5.1). Here we find its backward solution and an admissible set of final input and output conditions for a solution to exist.

```
 dalek%maple
    |\^/|     Maple V Release 3 (Loughborough University)
._|\|   |/|_. Copyright (c) 1981-1994 by Waterloo Maple Software and the
 \  MAPLE  /  University of Waterloo. All rights reserved. Maple and Maple V
 <____ ____>  are registered trademarks of Waterloo Maple Software.
      |       Type ? for help.
> with(linalg):
Warning: new definition for   norm
Warning: new definition for   trace
> read `back.m`;
> read `BACKADMISS.m`;
```

```
> A:=matrix(3,3,[s^2+5*s+6,s+1,0,2*s-5,3*s+2,1,0,-1,0]);

                              [  2                        ]
                              [ s  + 5 s + 6   s + 1   0  ]
                              [                           ]
                         A := [    2 s - 5    3 s + 2  1  ]
                              [                           ]
                              [       0          -1    0  ]

> B:=matrix(3,1,[0,0,1]);

                                        [ 0 ]
                                        [   ]
                                    B := [ 0 ]
                                        [   ]
                                        [ 1 ]

> BACK(A,B,2);

       [   - 1/6 ykplus21 - 5/6 ykplus11 - 1/6 ykplus12 + 1/6 ukplus01  ]
       [                                                               ]
       [                            - ukplus01                         ]
       [                                                               ]
       [ - 5/6 ykplus21 - 37/6 ykplus11 - 23/6 ykplus12 + 17/6 ukplus01 ]

> BACKADMISS(A,B,2,'L');

                              No restrictions apply

> quit
bytes used=679904, alloc=589716, time=0.73
dalek%
```

Here no restrictions apply on the final conditions and hence they can be chosen arbitrarily. However consider the matrix $A(\sigma)$ as given by

$$\begin{pmatrix} \sigma^2+5\sigma & \sigma+1 & 0 \\ 2\sigma & 3\sigma & 1 \\ 0 & -1 & 0 \end{pmatrix} \tag{5.4}$$

instead of $A(\sigma)$ in (5.1). We can therefore find the backward solution and admissible final conditions for this new system

```
dalek%maple
    |\^/|     Maple V Release 3 (Loughborough University)
._|\|   |/|_. Copyright (c) 1981-1994 by Waterloo Maple Software and the
 \  MAPLE  /  University of Waterloo. All rights reserved. Maple and Maple V
 <____ ____>  are registered trademarks of Waterloo Maple Software.
      |       Type ? for help.
> with(linalg):
Warning: new definition for   norm
Warning: new definition for   trace
> read `back.m`;
> read `BACKADMISS.m`;
> A:=matrix(3,3,[s^2+5*s,s+1,0,2*s,3*s,1,0,-1,0]);

                                [  2                ]
                                [ s  + 5 s  s + 1  0 ]
                                [                   ]
                           A := [    2 s     3 s   1 ]
                                [                   ]
                                [     0       -1   0 ]

> B:=matrix(3,1,[0,0,1]);

                                      [ 0 ]
                                      [   ]
                                  B := [ 0 ]
                                      [   ]
                                      [ 1 ]

> BACK(A,B,2);

        [ 1/25 ykplus21 + 1/25 ykplus12 + 4/25 ukplus01 + 1/5 ukmin11 ]
        [                                                             ]
        [                          - ukplus01                         ]
        [                                                             ]
        [         2/5 ykplus21 - 13/5 ykplus12 - 2/5 ukplus01         ]
```

```
> BACKADMISS(A,B,2,'L');

      [  1   0  0  5   1   0 ]  [ yNmin01 ]     [ 0  0   1  ]
      [                     ]  [          ]     [           ]
      [ 2/5  0  0  2  2/5  0 ]  [ yNmin02 ]     [ 0  0  2/5 ]
      [                     ]  [          ]     [           ]  [ uNmin01 ]
      [  0   0  0  0   0   0 ]  [ yNmin03 ]     [ 0  0   0  ]  [          ]
      [                     ], [          ], =, [           ], [ uNmin11 ]
      [ 1/5  0  0  1  1/5  0 ]  [ yNmin11 ]     [ 0  0  1/5 ]  [          ]
      [                     ]  [          ]     [           ]  [ uNmin21 ]
      [  0   0  0  0   0   0 ]  [ yNmin12 ]     [ 0  0   0  ]
      [                     ]  [          ]     [           ]
      [  0   0  0  0   0   0 ]  [ yNmin13 ]     [ 0  0   0  ]

> print(L);

                          [           _t[8]           ]
                          [                           ]
                          [           _t[6]           ]
                          [                           ]
                          [           _t[7]           ]
                          [                           ]
                          [           _t[5]           ]
                          [                           ]
                          [           _t[4]           ]
                          [                           ]
                          [           _t[3]           ]
                          [                           ]
                          [           _t[1]           ]
                          [                           ]
                          [           _t[2]           ]
                          [                           ]
                          [ - _t[8] - 5 _t[5] - _t[4] ]

> quit
bytes used=1243344, alloc=851812, time=1.35
dalek%
```

Here $L \in \mathbb{R}^9$ defines an admissible set of final input and output conditions. Infact L is of the form

$$\begin{pmatrix} y_1(N) \\ y_2(N) \\ y_3(N) \\ y_1(N-1) \\ y_2(N-1) \\ y_3(N-1) \\ -u(N) \\ -u(N-1) \\ -u(N-2) \end{pmatrix} \tag{5.5}$$

5.3 Symmetric Solution – Example

Consider the ARMA–Representation as given in (5.1). Here we find its symmetric solution for a given value of k and its boundary mapping equation which represents the restrictions placed on the initial and final outputs and the input sequence $u(k)$ of the system. Consider $N = 10$ i.e. the interval range is $[0,\ 10]$.

```
dalek%maple
    |\^/|     Maple V Release 3 (Loughborough University)
._|\|   |/|_. Copyright (c) 1981-1994 by Waterloo Maple Software and the
 \  MAPLE  /  University of Waterloo. All rights reserved. Maple and Maple V
 <____ ____>  are registered trademarks of Waterloo Maple Software.
      |       Type ? for help.
> with(linalg):
Warning: new definition for   norm
Warning: new definition for   trace
> read `sym.m`;
> read `SYMADMISS.m`;
> A:=matrix(3,3,[s^2+5*s+6,s+1,0,2*s-5,3*s+2,1,0,-1,0]);

                              [  2                        ]
                              [ s  + 5 s + 6   s + 1   0  ]
                              [                           ]
                         A := [    2 s - 5    3 s + 2  1  ]
                              [                           ]
                              [       0          -1    0  ]
```

```
> B:=matrix(3,1,[0,0,1]);
                                      [ 0 ]
                                      [   ]
                                B := [ 0 ]
                                      [   ]
                                      [ 1 ]
> SYMMETRIC(A,B,2,10,2);

           [         - 5 y11 - 6 y01 - 5 y02 + u11 - 4 u01        ]
           [                                                      ]
           [                        - u21                         ]
           [                                                      ]
           [ - 63 y11 - 90 y01 - 63 y02 + 3 u31 + 13 u11 - 48 u01 ]

> SYMMETRIC(A,B,2,10,3);

      [         19 y11 + 30 y01 + 19 y02 + u21 - 4 u11 + 14 u01         ]
      [                                                                 ]
      [                            - u31                                ]
      [                                                                 ]
      [ 225 y11 + 378 y01 + 225 y02 + 3 u41 + 13 u21 - 48 u11 + 162 u01 ]

> SYMMETRIC(A,B,2,10,7);

   [2059 y11 + 3990 y01 + 2059 y02 + u61 - 4 u51 + 14 u41 - 46 u31 + 146 u21

        - 454 u11 + 1394 u01]

                                   [- u71]

    [22905 y11 + 44658 y01 + 22905 y02 + 3 u81 + 13 u61 - 48 u51 + 162 u41

         - 522 u31 + 1638 u21 - 5058 u11 + 15462 u01]

> SYMMETRIC(A,B,2,10,8);

  [- 6305 y11 - 12354 y01 - 6305 y02 + u71 - 4 u61 + 14 u51 - 46 u41 + 146 u31

       - 454 u21 + 1394 u11 - 4246 u01]

                                   [- u81]

   [- 69867 y11 - 137430 y01 - 69867 y02 - 3 y92 + 13 u71 - 48 u61 + 162 u51

        - 522 u41 + 1638 u31 - 5058 u21 + 15462 u11 - 46962 u01]
```

```
> SYMADMISS(A,B,2,10,'L');

[  1  0  0   0   1  0   58025   0  0   115026   58025   0 ]  [ y101 ]
[                                                          ]  [      ]
[  0  0  0   0   0  0     0     0  0     0        0     0 ]  [ y102 ]
[                                                          ]  [      ]
[ 15  0  0  12  15  0   640323  0  0  1271430   640323  0 ]  [ y103 ]
[                                                          ]  [      ]
[  0  0  0   1   0  0   -19171  0  0   -37830   -19171  0 ]  [  y91 ]
[                                                          ]  [      ]
[  0  0  0   0   0  0     0     0  0     0        0     0 ]  [  y92 ]
[                                                          ]  [      ]
[ -2  0  0   5  -2  0  -211905  0  0  -419202  -211905  0 ]  [  y93 ]
[                                                          ], [      ], =,
[  0  0  0   0   0  0     0     0  0     0        -1    0 ]  [  y11 ]
[                                                          ]  [      ]
[  0  0  0   0   0  0     0     1  0     0        0     0 ]  [  y12 ]
[                                                          ]  [      ]
[  0  0  0   0   0  0    -15    0  1    -12       -15    0 ]  [  y13 ]
[                                                          ]  [      ]
[  0  0  0   0   0  0     0     0  0     0        0     0 ]  [  y01 ]
[                                                          ]  [      ]
[  0  0  0   0   0  0     0     0  0     0        1     0 ]  [  y02 ]
[                                                          ]  [      ]
[  0  0  0   0   0  0     2     0  0     -5       2     1 ]  [  y03 ]

[ 0  0   -4   14   -46   146   -454   1394   -4246   12866   -38854 ]  [ u101 ]
[                                                                   ]  [      ]
[ 0  0   0    0     0     0     0      0       0       0       0    ]  [  u91 ]
[                                                                   ]  [      ]
[ 0  0  -48  162  -522  1638  -5058  15462  -46962  142038  -428418 ]  [  u81 ]
[                                                                   ]  [      ]
[ 0  0   1    -4   14    -46   146    -454   1394    -4246   12866  ]  [  u81 ]
[                                                                   ]  [      ]
[ 0  0   0    0     0     0     0      0       0       0       0    ]  [  u71 ]
[                                                                   ]  [      ]
[ 0  0   13  -48   162  -522   1638  -5058   15462  -46962   142038 ]  [  u61 ]
[                                                                   ], [      ]
[ 0  0   0    0     0     0     0      0       0       0       1    ]  [  u51 ]
[                                                                   ]  [      ]
[ 0  0   0    0     0     0     0      0       0      -1       0    ]  [  u41 ]
[                                                                   ]  [      ]
[ 0  0   0    0     0     0     0      0       3       0       13   ]  [  u31 ]
[                                                                   ]  [      ]
[ 0  0   0    0     0     0     0      0       0       0       0    ]  [  u21 ]
[                                                                   ]  [      ]
```

```
[ 0  0   0    0     0     0     0      0       0       0       -1   ]  [  u11 ]
[                                                                    ]  [      ]
[ 0  0   0    0     0     0     0      0       0       3       0    ]  [  u01 ]
```

```
> print(L);
```

```
   [- _t[3] - 58025 _t[5] - 115026 _t[6] - 19171 _t[17] + 4 _t[9] - 14 _t[10]

        + 46 _t[11] - 146 _t[12] + 454 _t[13] - 1394 _t[14] + 4246 _t[15]

        - 12866 _t[16]]

                                    [_t[1]]

                                    [_t[2]]

    [19171 _t[5] + 37830 _t[6] + 6305 _t[17] - _t[9] + 4 _t[10] - 14 _t[11]

         + 46 _t[12] - 146 _t[13] + 454 _t[14] - 1394 _t[15] + 4246 _t[16]]

                                    [_t[3]]

                                    [_t[4]]

                                    [_t[5]]

                                    [_t[16]]

                  [15 _t[5] + 12 _t[6] + 2 _t[17] - 3 _t[15]]

                                    [_t[6]]

                                    [_t[17]]

                  [- 2 _t[5] + 5 _t[6] - 2 _t[17] - 3 _t[16]]

                                    [_t[7]]

                                    [_t[8]]

                                    [_t[9]]

                                    [_t[10]]

                                    [_t[11]]
```

```
[_t[12]]

[_t[13]]

[_t[14]]

[_t[15]]

[_t[16]]

[_t[17]]

> quit
bytes used=4761868, alloc=1441528, time=6.92
dalek%
```

Here we have selected k to have the values 2, 3, 7 and 8. In the above $L \in \mathbb{R}^{23}$ defines an admissible set of initial and final input and output conditions. Infact L is of the form

$$\begin{pmatrix} y_1(N) \\ y_2(N) \\ y_3(N) \\ y_1(N-1) \\ y_2(N-1) \\ y_3(N-1) \\ y_1(1) \\ y_2(1) \\ y_3(1) \\ y_1(0) \\ y_2(0) \\ y_3(0) \\ -u(10) \\ \vdots \\ -u(0) \end{pmatrix} \tag{5.6}$$

6 Conclusions

In this paper we have implemented the results presented in [3], concerned with the forward, backward and symmetric solution of an ARMA–Representation, in the symbolic computational language MAPLE. We have also implemented the corresponding compatibility conditions for each such solution which define an admissible set of boundary conditions, in conjunction with the input sequence, for such a solution to exist. All the corresponding MAPLE code has been included and a concise illustrative example given to show how these procedures are used and to show the simplicity of the whole implementation.

(**NOTE:** The MAPLE code presented, and several other procedures connected to the solution of ARMA–Representations, may be obtained by contacting N. Karampetakis at karampet@auth.gr or at the above correspondence address.)

References

[1] Char, B. W., Geddes, K. G., Gonnet, G. H. and Watt, S. M. (1991). *Maple V Language Reference Manual.* Springer-Verlag.

[2] Fragulis, G., Mertzios, B. G. and Vardulakis, A. I. (1991). Computation of the inverse of a polynomial matrix and evaluation of its Laurent expansion. *Int. J. Control* **53** 431–443.

[3] Karampetakis, N. P., Jones, J. and Antoniou, S. (2001). Forward, backward and symmetric solutions of discrete time ARMA–Representations. *Circuits, Systems & Signal Processing* **20** 89–109.

[4] Lewis, F. L. and Mertzios, B. G. (1990). On the Analysis of Discrete Linear Time–Invariant Singular Systems. *IEEE Trans. Automat. Contr.* **35** 506–511.

[5] Mertzios, B. G. (1984). Leverrier's Algorithm for Singular Systems. *IEEE Trans. Automat. Contr.* **29** 652–654.