Polynomial Toolbox Version 1.6

Tutorial

June 30, 1998

© Didier Henrion, Martin Hromcik, Huibert Kwakernaak, Sonja Pejchová, Michael Sebek, Rens C. W. Strijbos
Tutorial table of contents

Polynomial Toolbox Version 1.6

Tutorial

Contents

Quick start
Data structure
Numerical methods for polynomial matrices
General
Glossary
References
Quick start

Input a polynomial matrix

To input a polynomial matrix such as
\[ P(s) = \begin{bmatrix} 1+s & -s \\ s & 1-2s^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} s + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} s^2 \]

first input its coefficient matrices and degree

```matlab
» P0 = [1 0; 0 1];
» P1 = [1 -1; 1 0];
» P2 = [0 0; 0 -2];
» degP = 2;
```

Next, juxtapose the coefficient matrices and pack the matrix in polynomial matrix format

```matlab
» P = [P0 P1 P2];
» P = ppck(P,degP);
```

View a polynomial matrix

Inspect the result

```matlab
» P

P =
 1 0 1 -1 0 0 2
 0 1 1 0 0 -2 0
 0 0 0 0 0 0 NaN
```

The NaN in the bottom right corner indicates that \( P \) represents a polynomial matrix. The degree 2 has gone into the top right corner.

You get a better view of the polynomial matrix by displaying its coefficient matrices

```matlab
» pshow(P)

(2 x 2) polynomial matrix of degree 2.

Coefficient matrix at power 0
   1 0
   0 1
```
Coefficient matrix at power 1
1  -1
1   0

Coefficient matrix at power 2
0   0
0  -2

The best way to see small polynomial matrices is to use the `pdp` command:

```matlab
» pdp(P)
```

Columns 1 through 2

\[
\begin{array}{cc}
1 + s & -s \\
s & 1 - 2s^2
\end{array}
\]

Compute the determinant of a polynomial matrix

Now compute the determinant of \( P \)

```matlab
» pdet(P)
```

```
ans =
1   1  -1  -2   3
0   0   0   0   NaN
```

The determinant is a polynomial of degree 3 (see the top right corner). Its coefficients appear in the first row. We see that the determinant is

\[
\det(P(s)) = 1 + s - s^2 - 2s^3
\]

Compute the roots of a polynomial matrix

The roots of \( P \), that is, the roots of the determinant, are also easily computed

```matlab
» proots(ans)
```

```
ans =
-0.6647 + 0.4011i
-0.6647 - 0.4011i
 0.8295
```

We may compute the roots directly from the polynomial matrix \( P \) (by a different, possibly more reliable algorithm than via the determinant) as

```matlab
» proots(P)
```

```
ans =
-0.6647 + 0.4011i
-0.6647 - 0.4011i
 0.8295
```

Back to the main page
Polynomial matrix data structure

A polynomial matrix is a matrix whose entries are polynomials in an indeterminate variable, which we shall always denote as $s$. Alternatively, a polynomial matrix may be viewed as a polynomial whose coefficients are matrices. Thus we have

$$P(s) = \begin{bmatrix} 1 + s & 2 & s + s^2 \\ -s & 2s & 3 - s^2 \\ 1 + s + s^4 & 1 + 5s^2 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 3 \\ 1 & 1 & 8 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} s + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} s^2 + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} s^3 + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} s^4$$

More generally we express a polynomial matrix as

$$P(s) = P_0 + P_1 s + \cdots + P_n s^n$$

with $n$ the degree of the polynomial matrix. To store the matrix we need to save the coefficients matrices. Because the only data structure that MATLAB version 4 knows is a matrix the Toolbox packs the coefficient matrices side by side in a single matrix like this

$$P = \begin{bmatrix} P_0 & P_1 & \cdots & P_n \end{bmatrix}$$

This way of string the matrix, however, leaves the degree and the num columns ambiguous. A 3×15 packed matrix may represent a 3×3 polynomial matrix of degree 4, but also a 3×5 polynomial matrix of degree 2. To resolve this ambiguity an extra row and column are added, with the degree of the polynomial matrix in the top position of the extra column. The bottom entry of the extra column carries the NaN (“not a number”) symbol to identify the matrix as a polynomial matrix. Thus, the polynomial matrix of the example is represented in MATLAB as

$$P = \begin{bmatrix} 1 & 2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & 0 & -1 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 8 & 1 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & \text{NaN} \end{bmatrix}$$

The partitioning is not stored by MATLAB but inferred from the degree the top right corner and the size of the matrix.

The implementation of some macros requires matrices with a negative For this special situation we define the following format. Polynomials (polynomial matrices may have a negative degree equal to $-\infty$ (called MATLAB), but only if they are nonempty and identical to zero.
We thus distinguish three types of matrices

- Constant matrices, stored in a regular MATLAB two-dimension structure
- Polynomial matrices, stored in an extended data structure as explained above
- Negative degree matrices in the sense of the definition above.

To identify a matrix a macro `pinfo.m` is needed. It returns the type of a MATLAB data structure besides information about the size.

**Commands**

The following commands are available to pack, unpack and identify a polynomial matrix

- `ppck` Pack a matrix as a polynomial matrix
- `punpck` Unpack a polynomial matrix
- `pinfo` Identify the type of MATLAB data structure

Revised on 1998-06-01
Numerical methods for polynomial matrices

M. Sebek

Introduction

The algorithms that are used in the Polynomial Toolbox use different types of numerical techniques. They may be classified as follows:

- methods based on equating indeterminate coefficients
- polynomial reduction based on elementary row and column operations
- interpolation methods
- state space methods
- other methods

To learn quickly how the first four of these methods work, scan some easy examples.

Example 1: Scalar linear polynomial equation

Consider solving the scalar polynomial equation of the form

\[ a(x)x(x) + b(x)y(x) = c(x) \]

with \( a, b \) and \( c \) given polynomials, for the unknown polynomials \( x \) and \( y \).

For simplicity we assume that \( \deg a = \deg b = \deg c = 2 \). Then whenever the equation is solvable there exists at least one solution \( x, y \) characterized by

\[ \deg x \leq 1, \quad \deg y \leq 1 \]

The equation may be solved by equating indeterminate coefficients, polynomial reduction or interpolation as described below. In the Polynomial Toolbox this job is done by macro \texttt{xaybc}.

Example 2: Determinant of a polynomial matrix

For a simple \( 2 \times 2 \) polynomial matrix of degree 2

\[ P(\sigma) = F_0 + F_1\sigma + F_2\sigma^2 \]

consider the computation of its determinant

\[ P(\sigma) = \det P(\sigma) = F_0 + F_1\sigma + F_2\sigma^2 + F_3\sigma^3 + F_4\sigma^4 \]

Note that \( F_4 \neq 0 \) if and only if \( P \) is nonsingular.

The determinant may be found by polynomial reduction, by interpolation or by state space methods as described below. In the Polynomial Toolbox the computation of the determinant is handled by the macro \texttt{pdet}.
Equating indeterminate coefficients

Example 1 continued: Scalar linear polynomial equation. Let us see how the scalar linear polynomial equation may be handled by equating indeterminate coefficients. We begin by writing

$$a(s) = a_0 + a_1 s + a_2 s^2$$
$$b(s) = b_0 + b_1 s + b_2 s^2$$
$$a(s) = c_0 + c_1 s + c_2 s^2$$

where the coefficients are all known. Likewise, we write

$$x(s) = x_0 + x_1 s$$
$$y(s) = y_0 + y_1 s$$

with the unknown coefficients to be determined.

Step 1. By expanding the expression $xa + yb$ and equating coefficients of like powers in the indeterminate variable $s$ we obtain a set of linear equations of the form

$$\begin{bmatrix} x_0 & x_1 \end{bmatrix} \begin{bmatrix} a_0 & a_1 & a_2 & 0 \\ b_0 & b_1 & b_2 & 0 \\ 0 & a_0 & a_1 & a_2 \\ 0 & b_0 & b_1 & b_2 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ 0 \end{bmatrix}$$

The matrix $S$ has a highly structured form and is called the Sylvester resultant matrix corresponding to the polynomials $a$ and $b$.

Step 2. Solve the constant matrix equation to obtain the unknown coefficients of the polynomials $x$ and $y$ and, hence, the polynomials themselves. If the set of linear equations is not solvable then the polynomial matrix has no solution either.

Equating indeterminate coefficients

Discussion. The procedure is quite natural. It applies whenever

- the degree of the expected solution is known, and
- the constant matrix system is linear, of a reasonable size and easily constructed. Then it may efficiently be solved by any standard numerically stable linear system solver.

The knowledge of the resulting degree is here crucial: It guarantees that the correspondence between the polynomial equation and the linear system is one-to-one.

If the degree is not known then it is necessary to proceed heuristically: Just try a large enough degree and check whether or not the resulting linear system is solvable. If it is then the desired polynomial solution has been found. But if it is not then nothing can be concluded. There may or may not exist polynomial solutions of higher degree.
In the Polynomial Toolbox, the equating indefinite coefficients methods is employed in equation solvers such as axb, axbyc, axybc, axxa2b.

For further reading see the references.

## Polynomial reduction

### Example 1 continued: Scalar linear polynomial equation. To solve the scalar linear polynomial equation by polynomial reduction we proceed as follows.

**Step 1.** Use the polynomials $x$ and $y$ to form the polynomial matrix

$$
\begin{bmatrix}
  a(s) & 1 & 0 \\
  b(s) & 0 & 1 
\end{bmatrix}
$$

Use elementary row operations to reduce the matrix until its lower left corner equals $0$. After completion the matrix has the form

$$
\begin{bmatrix}
P(s) & r(s) \\
0 & q(s) & t(s)
\end{bmatrix}
$$

The polynomial $g$ is the greatest common divisor of $a$ and $b$ while $p$, $q$, $r$ and $t$ are coprime polynomials that satisfy

$$
p(s)a(s) + q(s)b(s) = g(s) \\
r(s)a(s) + t(s)b(s) = 0
$$

**Step 2.** Extract $g$ from the right hand side polynomial $c$ to obtain a polynomial $\bar{c}$ so that

$$
c(s) = \bar{c}(s)g(s)
$$

If this is not possible then **stop** because the polynomial equation is unsolvable.

**Step 3.** Take

$$
\begin{align*}
  x(s) &= \bar{c}(s)p(s) \\
  y(s) &= \bar{c}(s)q(s)
\end{align*}
$$

to be the desired solution. Moreover, all solutions to the polynomial equation may be expressed as

$$
\begin{align*}
  x(s) &= \bar{c}(s)p(s) + t(s)u(s) \\
  y(s) &= \bar{c}(s)q(s) + t(s)u(s)
\end{align*}
$$

with $u$ an arbitrary free polynomial parameter.
Example 2 continued: Computation of the determinant. To compute the determinant of the polynomial matrix $P$ by polynomial reduction we proceed this way.

Step 1. Using elementary row operations, transform the given matrix $P$ into a lower triangular matrix of the form

$$
\begin{bmatrix}
q_1(x) & 0 \\
t_21(x) & t_{22}(x)
\end{bmatrix}
$$

Step 2. Because elementary operations preserve the determinant the desired result may immediately be calculated as

$$\text{det}(P(x)) = \text{det}(T(x)) = t_{11}(x)t_{22}(x)$$

---

Polynomial reduction

Discussion. This is a traditional method of real "polynomial flavor." A typical feature of this method is that no attention is paid to the degree of the polynomials, which may grow alarmingly during the computation.

Unfortunately, the method is not numerically stable and, if the given data are "bad" then the performance of the method heavily depends on effective zeroing.

Finally, the method turns out to be rather slow when programmed in MATLAB.

In the Polynomial Toolbox polynomial reductions are employed in the functions pstairs, hermite, smith, exfac, gld, axb, axbyc, axybc and others.

When running some of these macros you may enjoy watching animations activated by option 'movie'.

For further reading see the references.

---

Interpolation

Example 1 continued: Linear polynomial equation. The "interpolation way" to determine the unknown polynomials $x$ and $y$ from the equation $ax + by = c$ consists of the following steps.

Step 1. For the problem at hand, choose four distinct complex interpolation points

$\zeta_1, \zeta_2, \zeta_3, \zeta_4$

and substitute them into the polynomials $a, b$ and $c$ to obtain scalar constants

$$a(\zeta_i), b(\zeta_i), c(\zeta_i), \quad i = 1, 2, 3, 4$$

Step 2. Form the linear equation system
Numerical methods for polynomial matrices

Step 3. Solve the equation system for the desired coefficients of the polynomials \( x \) and \( y \).

\[
\begin{bmatrix}
[x_0 & y_0 & x_1 & y_1]
\end{bmatrix}
\begin{bmatrix}
a(s_1) & a(s_2) & a(s_3) & a(s_4) \\
b(s_1) & b(s_2) & b(s_3) & b(s_4) \\
s_1a(s_1) & s_2a(s_2) & s_3a(s_3) & s_4a(s_4) \\
s_1b(s_1) & s_2b(s_2) & s_3b(s_3) & s_4b(s_4)
\end{bmatrix}
= \begin{bmatrix}
c(s_1) & c(s_2) & c(s_3) & c(s_4)
\end{bmatrix}
\]

Interpolation

Example 2 continued. Computation of the determinant. Quite similarly the determinant may be computed by interpolation.

Step 1. For the problem at hand, choose five distinct interpolation points

\( \sigma_i, \quad i=1,2,\ldots,5 \)

substitute them into the given matrix \( P \) to obtain five constant matrices

\( P(\sigma_i), \quad i=1,2,\ldots,5 \)

Step 2. Calculate the determinants

\( p(\sigma_i) = \det P(\sigma_i), \quad i=1,2,\ldots,5 \)

Step 3. Recover the desired coefficients of

\( p(\sigma) = \det P(\sigma) = p_0 + p_1\sigma + p_2\sigma^2 + p_3\sigma^3 + p_4\sigma^4 \)

by solving the linear equation system

\[
\begin{bmatrix}
p_0 & p_1 & p_2 & p_3 & p_4
\end{bmatrix}
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
\sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 \\
\sigma_1^2 & \sigma_2^2 & \sigma_3^2 & \sigma_4^2 & \sigma_5^2 \\
\sigma_1^3 & \sigma_2^3 & \sigma_3^3 & \sigma_4^3 & \sigma_5^3 \\
\sigma_1^4 & \sigma_2^4 & \sigma_3^4 & \sigma_4^4 & \sigma_5^4
\end{bmatrix}
= \begin{bmatrix}
p(\sigma_1) & p(\sigma_2) & p(\sigma_3) & p(\sigma_4) & p(\sigma_5)
\end{bmatrix}
\]

The matrix \( V \) is called a Vandermonde matrix.

Interpolation

Discussion. This technique for polynomial matrices is quite modern and efficient. Apparently, the larger part of computation is done within constant matrices the more efficient the method.

Like other methods, it requires the resulting degrees to be correctly determined a priori. If no
justified guess is available then the method becomes quite heuristic. If an incorrect degree is assumed then neither does solvability of the linear equation system guarantee the existence of a solution to the polynomial solution nor implies unsolvability of the linear system the non-existence of a solution to the polynomial problem.

The Vandermonde matrix appearing in the linear equation system often is ill conditioned. In many cases, this does not matter as a special "Vandermonde solver" may be employed. If this is not possible then the condition number limits the problem size.

In the Polynomial Toolbox the interpolation method may be encountered in \texttt{pdet}, \texttt{axb}, \texttt{axbyc}, \texttt{axxa2bc} and \texttt{pjsf}.

For further reading see the references.

**State space methods**

**Example 2 continued. Computation of the determinant.** To compute the determinant an indirect method may be used based on "state space" notions from linear system theory.

**Step 1.** From the matrix coefficients of

\[ P(s) = P_0 + P_1 s + P_2 s^2 \]

form the pair of generalized block companion matrices

\[
E = \begin{bmatrix} I_2 & 0 \\ 0 & P_2 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & I_2 \\ -P_0 & -P_1 \end{bmatrix}
\]

This pair may be considered to define a descriptor system \( E \dot{x} = Ax \).

**Step 2.** Compute the generalized eigenvalues corresponding to the matrices \( E, A \), that is, the roots of \( \det(E\lambda - A) \). Remove the infinite roots and denote the remaining finite roots as

\[ \delta_1, \delta_2, \delta_3, \delta_4 \]

They equal the finite roots of \( p(s) \).

**Step 3.** Recover \( p(s) = \det P(s) \) from its finite roots using the formula

\[ p(s) = c(s - \delta_1)(s - \delta_2) \cdots (s - \delta_4) \]

**State space methods**

**Discussion.** There are state space counterparts to many polynomial problems. As quite advanced numerical procedures have been developed for state-space problems the detour via systems theory is sometimes rewarding.

In the Polynomial Toolbox, state-space-like methods may be found in \texttt{proot}, \texttt{pdet} and \texttt{pjsf}. 
For further reading see the references.
<table>
<thead>
<tr>
<th>Macro names</th>
<th>Heading</th>
<th>Usage</th>
<th>Tolerances</th>
<th>Method</th>
</tr>
</thead>
</table>

**Macro names**

Many macro names in the toolbox are prefixed with the character `p`. Where possible existing MATLAB names for corresponding non-polynomial operations follow the prefix character.

**Heading**

The heading of each m-file contains a short description of the macro and all input and output variables. It serves as a help message when invoking the help facility of MATLAB.

**Usage**

If in the call of a macro the wrong number of input variables is specified the execution of the macro stops and a message is returned indicating the proper calling convention of the macro.

**Tolerances**

If the proper functioning of a macro depends on certain tolerance values, the user may specify these values as input parameters to the macro. In case these input parameters are not given a value, default values are substituted depending on the value of the permanent variable `eps` in MATLAB.

**Method**

If there are more methods available to do a specific job, only one macro is provided that incorporates all the different methods. A choice of the method is possible by defining the appropriate input parameter `method`. The default value for this input parameter is the "overall best" algorithm available.

Revised on 1998-06-01
Polynomial matrix glossary

### Polynomial matrices

We review some definitions and basic facts related to polynomial matrices. A \( k \times m \) polynomial matrix is a matrix of the form

\[
\mathcal{F}(s) = \begin{bmatrix} \bar{F}_1 + s \bar{R}_1 + \cdots + s^n \bar{R}_n \end{bmatrix}
\]

where \( s \) is an indefinite variable (usually taking its values in the complex plane), \( \bar{F}_i, \bar{R}_i, \ldots, \bar{R}_n \) are constant matrices (also known as coefficient matrices). Usually, unless stated otherwise, we deal with real matrices whose coefficient matrices are real.

If \( \mathcal{F}_n \) is not the zero matrix then we say that \( \mathcal{F} \) has degree \( n \). If \( \mathcal{F}_n \) is the zero matrix then we say that \( \mathcal{F} \) is said to be monic.

#### Tall and wide

A polynomial or other matrix is **tall** if it has at least as many rows as columns.

#### Rank

A polynomial matrix \( \mathcal{F} \) has full column rank (or full normal column rank) if it has full column rank everywhere in the complex plane except at a finite number of points. Similar definitions hold for full row rank and full rank.

The normal rank of a polynomial matrix \( \mathcal{F} \) equals

\[
\max_{s \in \mathbb{C}} \text{rank} \mathcal{F}(s)
\]

Similar definitions apply to the notions of normal column rank and normal row rank everywhere in the complex plane except at a finite number of points.

A square polynomial matrix is **nonsingular** if it has full normal rank.

#### Row and column degrees

Let the elements of the \( k \times m \) polynomial matrix \( \mathcal{F} \) be

\[
p_{ij}, \quad i = 1, 2, \ldots, k, \quad j = 1, 2, \ldots, m
\]

Then the numbers

\[
\rho_i = \max_j \deg p_{ij}, \quad i = 1, 2, \ldots, k
\]

\[
\gamma_j = \max_i \deg p_{ij}, \quad j = 1, 2, \ldots, m
\]

are the **row** and the **column degrees** of \( \mathcal{F} \), respectively.

#### Leading

Suppose that \( \mathcal{F} \) has column and row degrees...
respectively.

The column leading coefficient matrix of $P$ is the constant matrix whose coefficient of the term with power $\gamma_j$ of the $(i,j)$ entry of $P$.

The row leading coefficient matrix of $P$ is the constant matrix whose coefficient of the term with power $\rho_j$ of the $(i,j)$ entry of $P$.

**Column and row reduced**

A polynomial matrix is column reduced if its column leading coefficient matrix has full column rank. It is row reduced if its row leading coefficient matrix has full row rank.

**Conjugate**

If $P$ is a polynomial matrix then its conjugate $P^*$ is the polynomial matrix $P^*(\sigma) = P^H(-\sigma)$

The superscript $H$ indicates the complex conjugate transpose.

**Para-Hermitian**

A square polynomial matrix $P$ is para-Hermitian if $P^* = P$.

**Diagonally reduced**

The $m \times m$ para-Hermitian polynomial matrix $P$ is diagonally reduced if there exist degrees $\delta_1, \delta_2, \ldots, \delta_m$ so that the diagonal leading coefficient matrix exists and is nonsingular. $D$ is the diagonal polynomial matrix $D(\sigma) = \text{diag} \left( \sigma^{\delta_1}, \sigma^{\delta_2}, \ldots, \sigma^{\delta_m} \right)$

**Roots**

The roots or zeros of a polynomial matrix $P$ are those points in the complex plane.

If $P$ is square then its roots are the roots of its determinant $\det P$, including.

**Primeness**

A polynomial matrix $P$ is left prime if it has full row rank everywhere. It is right prime if it has full column rank everywhere in the complex plane.

**Coprimeness**

The $N$ polynomial matrices $\bar{R}_1, \bar{R}_2, \ldots, \bar{R}_N$ with the same numbers of rows is left prime. If the $N$ polynomial matrices all have the same numbers of columns then they are right coprime if...
Unimodular

A square polynomial matrix $U$ is **unimodular** if its determinant $\det U$ is an invertible polynomial matrices. The inverse of a unimodular polynomial matrix is again a polynomial matrix.

Matrix pencil

Matrix pencils are matrix polynomials of degree 1, such as

$$P(s) = P_1 + sP_2$$

Matrix pencils are often represented as polynomial matrices of the special form

$$P(s) = sE - A$$

but we shall normally consider matrix pencils as general polynomial matrices.

Elementary row and column operations

There are three basic elementary row operations:

- multiplying a row by a nonzero constant, such as

  $$\begin{bmatrix} 1 & s \\ 2 & s^2 \end{bmatrix} \text{ multiply first row by } 2 \Rightarrow \begin{bmatrix} 3 & 3s \\ 2 & 2s^2 \end{bmatrix}$$

- interchanging two rows, such as

  $$\begin{bmatrix} 1 & s \\ 2 & s^2 \end{bmatrix} \text{ interchange first and second row} \Rightarrow \begin{bmatrix} 2 & s^2 \\ 1 & s \end{bmatrix}$$

- adding a polynomial multiple of one row to another, such as

  $$\begin{bmatrix} 1 & s \\ 2 & s^2 \end{bmatrix} \text{ multiply second row by } s \text{ and add the result to the first row} \Rightarrow \begin{bmatrix} 1 + 2s & s + s^3 \\ 2 & 2s^2 \end{bmatrix}$$

Elementary column operations are defined analogously.

Diophantine equations

The simplest type of linear scalar polynomial equation - called Diophantine equations - is

$$a(s)x(s) + b(s)y(s) = c(s)$$

The polynomials $a$, $b$ and $c$ are given while the polynomial $x(s)$ and $y(s)$ are to be found. The equation is solvable if and only if the greatest common divisor of $a$ and $b$ divides $c$. If $a$ and $b$ are coprime, the equation is solvable for any right hand side polynomial, including $c = 1$.

The Diophantine equation possesses infinitely many solutions whenever there is any (particular) solution, then the general solution is
\[ x = x_0 + b t \\
 y = y_0 - a t \]

where \( t \) is an arbitrary polynomial (the parameter) and \( \overline{a}, \overline{b} \) are coprime such that
\[ \frac{\overline{b}}{\overline{a}} = \frac{b}{a} \]

If the \( a \) and \( b \) themselves are coprime then one can naturally take
\[ \overline{a} = a, \quad \overline{b} = b \]

Among all the solutions of Diophantine equation there exists a unique characterized by
\[ \deg x < \deg \overline{b} \]

There is another (generally different) solution pair characterized by
\[ \deg y < \deg \overline{a} \]

The two solution pairs coincide only if
\[ \deg a + \deg b > \deg c \]

**Bézout equations**

A Diophantine equation with 1 on its right hand side is called a Bézout like
\[ ax + by = 1 \]

with \( a \) and \( b \) given polynomials and \( x \) and \( y \) unknown.

**Zeroing**

Theoretically, the degree of a polynomial
\[ p(s) = p_n s^n + \cdots + p_0 \]

is \( n \) whenever \( p_n \neq 0 \). In numerical computations, however, one often very small (much smaller than the other coefficients) yet non-zero.

By way of example, consider two simple polynomials
\[ f(s) = 2 + (1 + \varepsilon)s + s^2 \]
\[ g(s) = 1 + s + s^2 \]

where \( \varepsilon \) is almost (but not quite) zero. When computing the difference
\[ f(s) - g(s) = 1 + \varepsilon s \]

a question on its degree may arise. It is necessary to compare \( |\varepsilon| \) with
coefficients to decide whether or not the corresponding term should be completely deleted. This process is called zeroing. The performance of many algorithms for polynomial problems critically depends on the way zeroing is done, in particular when elements of a Sylvester resultant matrix correspond to the polynomials

\[ a(s) = a_0 + a_1 s + \cdots + a_m s^m \]

\[ b(s) = b_0 + b_1 s + \cdots + b_n s^n \]

is the \((m+n)\times(m+n)\) constant matrix

\[
\begin{bmatrix}
  a_0 & a_1 & \cdots & a_m & 0 & \cdots & 0 \\
  0 & a_0 & \cdots & a_{m-1} & a_m & 0 & \cdots \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  0 & \cdots & 0 & a_0 & a_1 & \cdots & a_m \\
  b_0 & b_1 & \cdots & b_{n-1} & b_n & 0 & \cdots \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
  0 & \cdots & 0 & b_0 & b_1 & \cdots & b_n
\end{bmatrix}
\]

The resultant matrix is nonsingular if and only if the polynomials \(a\) and \(b\) are coprime.

Consider polynomials \(a, b\) and \(c\) such that \(a = bc\). We say that \(b\) is a divisor of \(c\) and write \(b|a\). This is sometimes also stated as \(b\) divides \(a\). If a polynomial \(g\) divides both \(a\) and \(b\) then \(g\) is called a common divisor of \(a\) and \(b\). If the only common divisors of \(a\) and \(b\) are constant then \(a\) and \(b\) are coprime.

If a polynomial \(m\) is a multiple of both \(a\) and \(b\) then \(m\) is called a common multiple of \(a\) and \(b\). If, furthermore, \(m\) is a divisor of every common multiple of \(a\) and \(b\) then \(m\) is a least common multiple of \(a\) and \(b\).

Next consider now polynomial matrices \(A, B, C\) of compatible size. Say that \(B\) is a left divisor of \(A\) or \(A\) is a right multiple of \(B\).

If a polynomial matrix \(G\) is a left divisor of both \(A\) and \(B\) then \(G\) is called a greatest common left divisor of \(A\) and \(B\). If, furthermore, \(G\) is a right multiple of every common left divisor of \(A\) and \(B\) then \(G\) is a least common right multiple of \(A\) and \(B\).

Right divisors, left multiples, common right divisors, greatest common left multiples, and least common left multiples are similarly defined.
<table>
<thead>
<tr>
<th>term</th>
<th>operation</th>
<th>term</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>coefficient matrix</td>
<td>full normal rank</td>
<td>column degrees</td>
<td>full rank</td>
</tr>
<tr>
<td>column leading coefficient matrix</td>
<td>half diagonal degrees</td>
<td>column rank</td>
<td>left coprime</td>
</tr>
<tr>
<td>column reduced</td>
<td>left prime</td>
<td>conjugate</td>
<td>row reduced</td>
</tr>
<tr>
<td>coprime</td>
<td>matrix pencil</td>
<td>degree</td>
<td>monic</td>
</tr>
<tr>
<td>diagonal leading coefficient matrix</td>
<td>multiple</td>
<td>diagonally reduced</td>
<td>nonsingular</td>
</tr>
<tr>
<td>Diophantine equations</td>
<td>normal rank</td>
<td>divisor</td>
<td>para-Hermitian</td>
</tr>
<tr>
<td>Top</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References and bibliography


20. Kwakernaak H. (1990), "MATLAB macros for polynomial $H_\infty$ control system optimization," Memorandum No. 881, Faculty of Applied Mathematics, University of Twente.


22. Kwakernaak H. (1993a), "State space algorithms for polynomial matrix computations," Memorandum No. 1168, Faculty of Applied Mathematics, University of Twente.


24. Kwakernaak H. (1996), "Frequency domain solution of the standard $H_\infty$


Revised on 1998-06-01