Polynomial Toolbox Version 1.6

Tutorial

June 30, 1998

Polynomial Toolbox Version 1.6

Tutorial

Contents

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

Opening page

Tutorial

Demos

Commands
- Summary
- Grouped by function

Manual

Documentation

Animations

Quick start

Input a polynomial matrix

View a polynomial matrix

Compute the determinant of a polynomial matrix

Compute the roots of a polynomial matrix

Return to the main page

Input a polynomial matrix

Back to top

To input a polynomial matrix such as

$$P(s)=\begin{bmatrix}1+s & -s\\ s & 1-2s^2\end{bmatrix}=\underbrace{\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}}_{P_0}+\underbrace{\begin{bmatrix}1 & -1\\ 1 & 0\end{bmatrix}}_{P_1}s+\underbrace{\begin{bmatrix}0 & 0\\ 0 & -2\end{bmatrix}}_{P_2}s^2$$

first input its coefficient matrices and degree

```
» 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

```
» P = [P0 P1 P2];

» P = ppck(P,degP);
```

View a polynomial matrix

Back to top

Inspect the result

```
» 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

```
» 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:

```
» pdp(P)

Columns 1 through 2

    1 + s      - s
    s        1 - 2s^2
```

Compute the determinant of a polynomial matrix

Back to top

Now compute the determinant of `P`

```
» 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

Back to top

The roots of `P,` that is, the roots of the determinant, are also easily computed

```
» 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

```
» proots(P)

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

Back to the main page

Revised on 1998-06-01

Data structure

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*. Alternativel polynomial matrix may be viewed as a polynomial whose coefficients a matrices. Thus we have

$$P(s)=\begin{bmatrix} 1+s & 2 & s+s^2 \\ -s & 2s & 3-s^2 \\ 1+s+s^4 & 1+5s^2 & 8 \end{bmatrix}$$
$$=\begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 3 \\ 1 & 1 & 8 \end{bmatrix}+\begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}s+\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 5 & 0 \end{bmatrix}s^2+\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}s^3+\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

More generally we express a polynomial matrix as

$$P(s)=P_0+P_1s+\cdots+P_ns^n$$

with *n* the degree of the polynomial matrix. To store the matrix we nee save the coefficients matrices. Because the only data structure that MA version 4 knows is a matrix the Toolbox packs the coefficient matrices 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 numl columns ambiguous. A 3×15 packed matrix may represent a 3×3 polyn matrix of degree 4, but also a 3×5 polynomial matrix of degree 2. To r this ambiguity an extra row and column are added, with the degree of t polynomial matrix in the top position of the extra column. The bottom the extra column carries the NaN ("not a number") symbol to identify t matrix as a polynomial matrix. Thus, the polynomial matrix of the exan represented in MATLAB as

$$P=\left[\begin{array}{ccc|ccc|ccc|ccc|ccc|c} 1 & 2 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & -1 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 8 & 1 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathrm{NaN} \end{array}\right]$$

The partitioning is not stored by MATLAB but inferred from the degre 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 c polynomial matrices may have a negative degree equal to $-\infty$ (`-Inf` ir 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 exp
- 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

Top

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

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

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

$$\deg x \le 1, \quad \deg y \le 1$$

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 `xaybc`.

Example 2: Determinant of a polynomial matrix

For a simple 2×2 polynomial matrix of degree 2

$$P(s)=P_0+P_1s+P_2s^2$$

consider the computation of its determinant

$$p(s)=\det P(s)=p_0+p_1s+p_2s^2+p_3s^3+p_4s^4$$

Note that $p_4 \neq 0$ 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 `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

$$\begin{aligned} a(s) &= a_0 + a_1 s + a_2 s^2 \\ b(s) &= b_0 + b_1 s + b_2 s^2 \\ a(s) &= c_0 + c_1 s + c_2 s^2 \end{aligned}$$

where the coefficients are all know. Likewise, we write

$$\begin{aligned} x(s) &= x_0 + x_1 s \\ y(s) &= y_0 + y_1 s \end{aligned}$$

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

$$\begin{bmatrix} x_0 & y_0 & x_1 & y_1 \end{bmatrix} \underbrace{\begin{bmatrix} a_0 & a_1 & a_2 & 0 \\ b_0 & b_1 & b_2 & 0 \\ 0 & a_0 & a_1 & a_2 \\ 0 & b_0 & b_1 & b_2 \end{bmatrix}}_{S} = \begin{bmatrix} c_0 & c_1 & c_2 & 0 \end{bmatrix}$$

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

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

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

$$\begin{aligned} p(s)a(s)+q(s)b(s) &= g(s) \\ r(s)a(s)+t(s)b(s) &= 0 \end{aligned}$$

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

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

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

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

with *u* an arbitrary free polynomial parameter.

Polynomial reduction

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

$$\begin{bmatrix} t_{11}(s) & 0 \\ t_{21}(s) & t_{22}(s) \end{bmatrix}$$

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

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

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

$$s_1, s_2, s_3, s_4$$

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

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

Step 2. Form the linear equation system

$$\begin{bmatrix} x_0 & y_0 & x_1 & y_1 \end{bmatrix} \underbrace{\begin{bmatrix} a(s_1) & a(s_2) & a(s_3) & a(s_4) \\ b(s_1) & b(s_2) & b(s_3) & b(s_4) \\ s_1 a(s_1) & s_2 a(s_2) & s_3 a(s_3) & s_4 a(s_4) \\ s_1 b(s_1) & s_2 b(s_2) & s_3 b(s_3) & s_4 b(s_4) \end{bmatrix}}_{V} = \begin{bmatrix} c(s_1) & c(s_2) & c(s_3) & c(s_4) \end{bmatrix}$$

Step 3. Solve the equation system for the desired coefficients of the polynomials *x* and *y*.

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

$$s_i, \quad i = 1, 2, \cdots, 5$$

substitute them into the given matrix *P* to obtain five constant matrices

$$P(s_i), \quad i = 1, 2, \cdots, 5$$

Step 2. Calculate the determinants

$$p(s_i) = \det P(s_i), \quad i = 1, 2, \cdots, 5$$

Step 3. Recover the desired coefficients of

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

by solving the linear equation system

$$\begin{bmatrix} p_0 & p_1 & p_2 & p_3 & p_4 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ s_1 & s_2 & s_3 & s_4 & s_5 \\ s_1^2 & s_2^2 & s_3^2 & s_4^2 & s_5^2 \\ s_1^3 & s_2^3 & s_3^3 & s_4^3 & s_5^3 \\ s_1^4 & s_2^4 & s_3^4 & s_4^4 & s_5^4 \end{bmatrix}}_{V} = \begin{bmatrix} p(s_1) & p(s_2) & p(s_3) & p(s_4) & p(s_5) \end{bmatrix}$$

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 pdet, axb, axbyc, axxa2bc and 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 $\det(\lambda E - A)$. Remove the infinite roots and denote the remaining finite roots as

$$s_1, s_2, s_3, s_4$$

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

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

$$p(s) = c(s - s_1)(s - s_2)\cdots(s - s_4)$$

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 proot, pdet and pjsf.

For further reading see the references.

Opening page

Tutorial

Demos

Commands
- Summary
- Grouped by function

Manual

Documentation

Animations

General

Macro names Heading Usage Tolerances Method

Return to the main page

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

Opening page

Tutorial

Demos

Commands
Summary
Grouped by function

Manual

Documentation

Animations

Polynomial and polynomial matrix glossary

Polynomial matrices

Index | Top

We review some definitions and basic facts related to polynomial matr

A $k\times m$ polynomial matrix is a matrix of the form

$$P(s)=P_0+P_1s+\cdots+P_ns^n$$

where *s* is an indefinite variable (usually taking its values in the compl
constant matrices

$$P_0, P_1, \cdots, P_n$$

coefficient matrices. Usually, unless stated otherwise, we deal with *re*
whose coefficient matrices are real.

If P_n is not the zero matrix then we say that *P* has *degree n*. If P_n is t
said to be *monic*.

Tall and wide

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

Rank

Index | Top

A polynomial matrix *P* has *full column rank* (or full *normal column r*
rank everywhere in the complex plane except at a finite number of poi
hold for *full row rank* and *full rank*.

The *normal rank* of a polynomial matrix *P* equals

$$\max_{s\in C}\ \text{rank}\ P(s)$$

Similar definitions apply to the notions of normal column rank and no

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

Row and column degrees

Index | Top

Let the elements of the $k\times m$ polynomial matrix *P* be

$$p_{ij},\quad i=1,2,\cdots,k,\quad j=1,2,\cdots,m$$

Then the numbers

$$\rho_i=\max_j \deg p_{ij},\quad i=1,2,\cdots,k$$
$$\gamma_j=\max_i \deg p_{ij},\quad j=1,2,\cdots,m$$

are the *row* and the *column degrees* of *P*, respectively.

Leading

Suppose that *P* has column and row degrees

coefficient matrices

Index | Top

$$\rho_i, \quad i = 1, 2, \cdots, k$$
$$\gamma_j, \quad j = 1, 2, \cdots, m$$

respectively.

The *column leading coefficient matrix* of *P* is the constant matrix whc
coefficient of the term with power γ_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 ρ_j of the (*i*, *j*) entry of *P*.

Column and row reduced

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

Conjugate

Index | Top

If *P* is a polynomial matrix then its *conjugate P** is the polynomial ma

$$P^*(s) = P^H(-s)$$

The superscript *H* indicates the complex conjugate transpose.

Para-Hermitian

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

Diagonally reduced

Index | Top

The $m \times m$ para-Hermitian polynomial matrix *P* is *diagonally reduced* i
degrees $\delta_1, \delta_2, \cdots, \delta_m$ so that the *diagonal leading coefficient matrix*

$$\lim_{|s| \to \infty} D^{-1}(-s) P(s) D^{-1}(s)$$

exists and is nonsingular. *D* is the diagonal polynomial matrix

$$D(s) = \mathrm{diag}\,(s^{\delta_1}, s^{\delta_2}, \cdots, s^{\delta_m})$$

Roots

Index | Top

The *roots* or *zeros* of a polynomial matrix *P* are those points in the coı
rank.

If *P* is square then its roots are the roots of its determinant det *P*, inclu

Primeness

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

Coprimeness

Index | Top

The *N* polynomial matrices $P_1, P_2, \cdots, P_N$ with the same numbers of row

$$\begin{bmatrix} P_1 & P_2 & \cdots & P_N \end{bmatrix}$$

is left prime. If the *N* polynomial matrices all have the same numbers c
right coprime if

$$\begin{bmatrix} P_1 \\ P_2 \\ \cdots \\ P_N \end{bmatrix}$$

is right prime.

Unimodular

A square polynomial matrix U is *unimodular* if its determinant det U i
inverse of a unimodular polynomial matrix is again a polynomial matri

Matrix pencil

Index | Top

Matrix pencils are matrix polynomials of degree 1, such as

$$P(s) = P_0 + P_1 s$$

Matrix pencils are often represented as polynomial matrices of the spe

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

but we shall normally consider matrix pencils as general polynomial m

Elementary row and column operations

Index | Top

There are three basic elementary row operations:

- multiplying a row by a nonzero constant, such as

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

- interchanging two rows, such as

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

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

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

Elementary column operations are defined analogously.

Diophantine equations

Index | Top

The simplest type of linear scalar polynomial equation - called Diopha
Alexandrian mathematician Diophantos (A.D. 275) is

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

The polynomials polynomials a, b and c are given while the polynomia
The equation is solvable if and only if the greatest common divisor of
implies that with relatively a and b coprime the equation is solvable fo
polynomial, including c = 1.

The Diophantine equation possesses infinitely many solutions whenev
is any (particular) solution, then the general solution is

$$x = x_o + \bar{b}t$$
$$y = y_o - \bar{a}t$$

where *t* is an arbitrary polynomial (the parameter) and $\bar{a}, \bar{b}$ are coprim

$$\frac{\bar{b}}{\bar{a}} = \frac{b}{a}$$

If the *a* and *b* themselves are coprime then one can naturally take

$$\bar{a} = a, \quad \bar{b} = b$$

Among all the solutions of Diophantine equation there exists a unique
characterized by

$$\deg x < \deg \bar{b}$$

There is another (generally different) solution pair characterized by

$$\deg y < \deg \bar{a}$$

The two solution pairs coincide only if

$$\deg a + \deg b \geq \deg c$$

Bézout equations

[Index](#) | [Top](#)

A Diophantine equation with 1 on its right hand side is called a Bézou
like

$$ax + by = 1$$

with *a* and *b* given polynomials and *x* and *y* unknown.

Zeroing

[Index](#) | [Top](#)

Theoretically, the degree of a polynomial

$$p(s) = p_0 + p_1 s + \cdots + p_n s^n$$

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 ε is almost (but not quit) 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 b
process is called *zeroing*. The performance of many algorithms for pol
critically depends on the way zeroing is done, in particular when elem

Sylvester resultant matrix

Index | Top

The Sylvester resultant matrix corresponding 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

$$S(a,b) = \begin{bmatrix} a_0 & a_1 & \cdots & a_m & 0 & \cdots & 0 \\ 0 & a_0 & \cdots & a_{m-1} & a_m & 0 & \cdots \\ \vdots & \cdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_0 & a_1 & \cdots & a_m \\ b_0 & b_1 & \cdots & b_n & 0 & \cdots & 0 \\ 0 & b_0 & \cdots & b_{n-1} & b_n & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & b_0 & b_1 & \cdots & b_n \end{bmatrix}$$

The resultant matrix is nonsingular if and only if the polynomials *a* anc

Divisors and multiples

Index | Top

Consider polynomials *a*, *b* and *c* such that *a* = *bc*. We say that *b* is a d
multiple of *b*, and write *b*|*a*. This is sometimes also stated as *b* divides

If a polynomial *g* divides both *a* and *b* then *g* is called a common divis
furthermore, *g* is a multiple of every common divisor of *a* and *b* then *g*
divisor of *a* and *b*. If the only common divisors of *a* and *b* are constan
and *b* are coprime.

If a polynomial *m* is a multiple of both *a* and *b* then *m* is called a comn
furthermore, *m* is a divisor of every common multiple of *a* and *b* then
multiple of *a* and *b*.

Next consider now polynomial matrices *A*, *B* and *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 cal
of *A* and *B*. If, furthermore, *G* is a right multiple of every common left
is a greatest common left divisor of *A* and *B*. If the only common left
unimodular matrices then the polynomial matrices *A* and *B* are left cop

If a polynomial matrix *M* is a right multiple of both *A* and *B* then *M* is
multiple of *A* and *B*. If, furthermore, *L* is a left divisor of every comm
right multiple of *A* and *B* then *G* is a least common right multiple of *A*

Right divisors, left multiples, common right divisors, greatest commor
left multiples, and least common left multiples are similarly defined.

Index

Bézout equations
elementary row and column
prime

coefficient matrix

column degrees

column leading coefficient matrix

column rank

column reduced

conjugate

coprime

degree

diagonal leading coefficient matrix

diagonally reduced

Diophantine equations

divisor

operations

full normal rank

full rank

half diagonal degrees

left coprime

row reduced

left prime

matrix pencil

monic

multiple

nonsingular

normal rank

para-Hermitian

rank

right

roots

row d

row le

row ra

Sylve

tall

unimo

zeroin

Top

Opening page

Tutorial

Demos

Commands
Summary
Grouped by function

Manual

Documentation

Animations

References and bibliography

1. Antsaklis P. J. and Z. Gao (1992), "Polynomial and rational matrix interpolation: Theory and control application." Control Systems Technical Report No. 71, University of Notre Dame, Indiana, U.S.A.
2. Barnett S. (1983), *Polynomial and Linear Control Systems*, Pure and Applied Mathematics, Marcel Dekker.
3. Callier F. M. (1985), "On polynomial matrix spectral factorization by symmetric factor extraction", *IEEE Trans. Aut. Control* **30**, pp. 453-464.
4. Callier F. M. and C. A. Desoer (1982), *Multivariable Feedback Systems*, Springer-Verlag, New York.
5. Chen Ch. (1984), *Linear System Theory and Design*, Holt, Rinehart and Winston, New York.
6. Datta K. B. and S. Gangopadhyay (1992), "Coprime matrix fraction description via orthogonal structure theorem," *IEEE Trans. Aut. Control*, **37,** pp. 1517-1520.
7. Dorf, R. C. (1989), *Modern Control Systems*, 5th Ed. Addison-Wesley.
8. Gantmacher F. R. (1960), *The Theory of Matrices*, Chelsea Publishing Company, New York.
9. Gohberg I., P. Lancaster and I. Rodman (1982), *Matrix Polynomials*, Academic Press, New York.
10. Golub G. H. and C. F. Van Loan (1989), *Matrix Computations*, John Hopkins University Press, second edition.
11. Henrion D. and M. Sebek (1996), "Symmetric matrix polynomial equations." Submitted.
12. Henrion D. and M. Sebek (1997), "An efficient numerical algorithm for the discrete time symmetric matrix polynomial equation." *Proc. 1997 European Control Conference*, Brussels.
13. Jakubovic V. A. (1970), "Factorization of symmetric matrix polynomials", *Dokl. Acad. Nauk. SSSR*, **194**, 3, pp. 1261-1264.
14. Jezek J. and V. Kucera (1985), "Efficient Algorithm for Matrix Spectral Factorization", *Automatica*, Vol. 21, No. 6, pp 663-669.
15. Kailath T. (1980), *Linear Systems*, Prentice Hall, Englewood Cliffs, N.J.
16. Kraffer, F., M. Sebek and S. Pejchová (1994), "Numerical performance in matrix Diophantine equation", *Kybernetika* **30**, 6, pp. 745-753.
17. Kucera V. (1979), *Discrete Linear Control*, New York: Wiley.
18. Kucera, V. (1991), *Analysis and Design of Discrete Linear Control Systems*, Academia and Prentice Hall, Prague and London.
19. Kucera V. (1993), "Diophantine equations in control - a survey", *Automatica* **29**, 6, pp. 1361-1375.
20. Kwakernaak H. (1990), "MATLAB macros for polynomial H_∞ control system optimization," Memorandum No. 881, Faculty of Applied Mathematics, University of Twente.
21. Kwakernaak H. (1991), "The polynomial approach to H_∞-optimal regulation", in E. Mosca and L. Pandolfi, Eds., H_∞ *Control Theory*, Lecture Notes in Mathematics **1496**, Springer-Verlag, Berlin.
22. Kwakernaak H. (1993a), "State space algorithms for polynomial matrix computations," Memorandum No. 1168, Faculty of Applied Mathematics, University of Twente.
23. Kwakernaak H. (1993b), "Robust control and H_∞ optimization," *Automatica*, **29**, pp. 255-273.
24. Kwakernaak H. (1996), "Frequency domain solution of the standard H_∞

problem." In M. J. Grimble and V. Kucera, Eds., *Polynomial Methods for Control Systems Design*, Springer-Verlag.

25. Kwakernaak H. and R. Sivan (1971), *Linear Optimal Control Systems.* Wiley-Interscience, New York.
26. Kwakernaak H. and M. Sebek (1994), "Polynomial *J*-Spectral Factorization", *IEEE Transactions on Automatic Control*, Vol. 39, No. 2, pp 315-328.
27. Patel R. V. (1981), "Computation of matrix fraction descriptions of linear time-invariant systems," *IEEE Trans. Aut. Control,* **26**, pp. 148-161.
28. Rosenbrock H. H. (1970), *State space and multivariable theory*. Wiley-Interscience, New York.
29. Sebek M. (1981), *Discrete Regulation and Tracking: An Algebraic Approach* (in Czech), Ph.D. thesis, Czechoslovak Academy of Sciences, Prague, Czechoslovakia.
30. Sebek, M. (1993), *"J*-Spectral factorisation algorithms", in ed. K. J. Hunt, *Polynomial Methods in Optimal Control and Filtering*, Ch. 10, pp. 278-307. Peter Peregrinus, Ltd, London.
31. Sebek M., S. Pejchová and F. Kraffer (1994), "Testing algorithms for linear polynomial equations", *Proceedings of the 33rd IEEE Conference on Decision and Control*, IEEE Control Systems Society, pp. 2518-2519, Orlando, Fl.
32. Sebek M., F. Kraus, S. Pejchová, H. Kwakernaak and D. Henrion (1996), "Numerical methods for zeros and determinant of polynomial matrix." *Proc. 4th IEEE Mediterranean Symposium on Control and Automation*, Chania, Crete, Greece, June 1996.
33. Sebek, M., D. Henrion, S. Pejchová and H. Kwakernaak (1997), "Numerical methods for zeros and determinant of polynomial matrix", *Kybernetika* (to appear).
34. Sebek M. and P. Husek (1997), "Basic routines for rational matrices: a critical comparison", *Proceedings of the 5th IEEE Mediterranean Symposium on New Directions in Control and Automation*, Paphos, Cyprus.
35. T. Söderström, J. Jezek and V. Kucera (1997), "An efficient and versatile algorithm for computing the covariance function of an ARMA process." Accepted for publication in *IEEE Trans. Signal Processing.*
36. Stefanidis P., A. P. Paplinski and M. J. Gibbard (1992), *Numerical Operations with Polynomial Matrices : Application to Multi-Variable Dynamic Compensator Design*. Lecture Notes in Control and Information Sciences, Vol. 171, Springer-Verlag, 1992.
37. Strijbos R. C. W. (1995a). "Calculation of right matrix fraction descriptions; an algorithm," Memorandum No. 1287, Faculty of Applied Mathematics, University of Twente.
38. Strijbos R. C. W. (1995b), "A polynomial toolbox; implementation aspects," Memorandum No. 1288, Faculty of Applied Mathematics, University of Twente.
39. Strijbos R. W. C. (1996), "Calculation of right matrix fraction descriptions; an algorithm," *Proceedings of the 4th IEEE Mediterranean Symposium on New Directions in Control and Automation*, Maleme, Krete, Greece, June 10-13, pp. 478-482.
40. Van der Waerden, B. L. (1966), *Modern Algebra*, Frederic Ungar Publishing Co., New York. Volumes I and II, Sixth Edition.
41. Wolovich W. A. (1974)}, *Linear Multivariable Systems*, Springer, New York.

42. Wolowich W. A. (1978), "Skew prime polynomial matrices", *EEE Trans. Aut. Control* **23**, pp. 880-887.

Revised on 1998-06-01