SOLUTION OF SIMPLE DIOPHANTINE EQUATIONS BY MEANS OF MATLAB

Vladimír Hanta

Institute of Chemical Technology, Department of Computing and Control Engineering

Abstract

At present, teaching in automatic control theory courses is carried out with a strong computer support usually in MATLAB environment. Polynomial Toolbox has been developed for operations on polynomials and polynomial matrices. An older version of that tool could be downloaded as freeware from the Internet. A new version of it, Polynomial Toolbox 2.5, is now available as a commercial product at a cost, which is unacceptable for intermittent use in the framework of teaching.
The contribution is devoted to taking full advantage of standard MATLAB commands for solving simple Diophantine equations by means of different methods. There are described three basic methods for solution of polynomial Diophantine equations (the Euclidean algorithm – polynomial division algorithm, method based on elementary transformations of matrices and method of indefinite coefficients). They all are soluble with support of MATLAB or its standard toolbox commands. The contribution also compares their computational complexity and characterizes them from the standpoint of demands put on inexpert users.

1 Introduction

Beside classical methods of design in frequency-, state- or time-domain, algebraic methods belong to the basic theoretical methods of the design of linear discrete control systems. Algebraic theory of discrete linear control has started developing since the seventies of the last century. Its essential advantage in contrast to classical methods is the fact that it is based on input-output description of a discrete system. Discrete controllers can be realized as a suitable difference equation only. With a good description of a system and with known input variables, control algorithms that are capable of error signal elimination in a finite number of control steps can be designed contrary to the continuous control methods of design.

2 Algebraic Control Methods in Education

Algebraic methods of control are standard parts of courses in the fields of automation and control engineering at technical universities. Tasks of discrete algebraic control synthesis used to be difficult to solve not long ago in the courses. All calculations, which were often complicated, were carried out without computer support. Nowadays, MATLAB computational and visualization system, has become a standard tool for control theory teaching (and also for a number of other scientific branches). Polynomial Toolbox program package for operations with polynomials and polynomial matrices involving tools for solutions of tasks of algebraic discrete control has been developed. This toolbox, which contains more than two hundred functions, used to be free to download from Internet.

A new object oriented version, which appeared in 1998, is completely a standard commercial product. The price of the Classroom Kit for schools is all of 500 dollars. Considering the need of the Polynomial Toolbox in control theory courses at ICHT Prague lasting about two weeks yearly only, its price is too high. In addition, synthesis of single- input single-output discrete control circuits is carried out in the courses only. These tasks are soluble in a simple way by means of the standard MATLAB and its parts, Control System Toolbox and Symbolic Toolbox.

3 Linear Diophantine Polynomial Equations

The linear Diophantine polynomial equation is indeterminate equation given in the form

$$ax + by = c$$

where a, b, c are given and x, y are unknown polynomials. A general solution of the equation can be found as a linear combination of a particular solution of the complete and a general solution of the reduced Diophantine equations:

$$\begin{aligned} x &= pc/d + rt \\ y &= qc/d + st \end{aligned}$$

where d is the greatest common divisor of the polynomials a, b and t is an arbitrary polynomial by means of which properties of a solution are set up.

4 Methods of Diophantine Equation Solving

In the framework of algebraic control theory teaching at ICHT in Prague, three methods that can be realized in the MATLAB and standard toolboxes are used for solving of simple Diophantine equations:

1. Euclidean algorithm for searching of the greatest common divisor of two polynomials,
2. method of elementary modifications of a matrix,
3. method of indefinite coefficients.

4.1 Euclidean Algorithm

Euclidean algorithm for searching of the greatest common divisor of two polynomials can be extended for solving :

$$\begin{aligned} h_0 &= a \\ h_1 &= b \\ h_0 &= q_1 h_1 + h_2 \\ h_1 &= q_2 h_2 + h_3 \\ &\dots \\ h_{m-2} &= q_{m-1} h_{m-1} + h_m \\ h_{m-1} &= q_m h_m + 0 \\ d &= (a, b) = h_m \end{aligned}$$

The extension of the algorithm for computation of the coefficients p, q for representation of the largest common divisor of two polynomials as their linear combination is presented below:

procedure Particular solution of a Diophantine equation $ax + by = c$ over a domain of integrity by Euclidean algorithm
begin
    input(a, b, c);
    $\alpha := a$; $\beta := b$; $i := 0$;
    repeat
        $i := i + 1$;
        $h_i :=$ quotient(α, β);
        $r :=$ remainder(α, β);
        $\alpha := \beta$; $\beta := r$;
    until $r = 0$;
    $d := \alpha$;
    $p := 0$; $q := 1$;
    for j **:=** $m - 1$ **step** -1 **to** 1 **do**
        begin
            $r := p$; $p := q$; $q := r - h_i * p$
        end;
    output(d, p, q)
end;

The source text of the corresponding m–file is as follows:

```
function [d,p,q,r,s] = euklidalg(a,b,c);
% Solution of Diophantine equation ax + by = c
% by means of Euclidean algorithm
% The greatest common divisor is the last nonzero remainder
% Back substitution:
% coefficients of the linear combination p, q
% r = -b/d, s = a/d
%
% test of equation correctness:
if isempty(find(a)) | isempty(find(b))
    display('Diophantine equation is not given correctly!')
    p=0; q=0; d=0; s=0; r=0;
    return
end
alfa=a; beta=b; rem=a;
% search for the greatest common divisor:
i=0;
n=abs(length(a)-length(b))+1;
if n == 1
    n=n+1;
end
% test of zero remainder:
while norm(rem,inf) > 100*eps
    i=i+1;
    % elimination the zero leading coefficients:
    ind=find(beta);
    beta=beta(ind(1):length(beta));
    % quotient and remainder:
    [quot,rem]=deconv(alfa,beta);
```

```
    i0=1+n-length(quot);
    % storing of quotients:
    qq(i,i0:n)=quot;
    % shift of polynomials:
    alfa=beta; beta=rem;
end
% recurrent computation of the coefficients
d=alfa; p=0; q=1; m=i-1
for i=m:-1:1
    r=p; p=q
    % formal rearrangement for polynomial sum executing:
    rr=zeros(1,length(qq(i,:))+length(p)-1);
    rr(length(rr)-length(r)+1:length(rr))=r;
    % computation of further element of the sequence:
    q=rr-conv(qq(i,:),p);
end
% normalization of polynomial:
ind=find(q); q=q(ind(1):length(q));
% general solution of the reduced equation:
r=-deconv(b,d); s=deconv(a,d);
return
```

4.2 Elementary modifications of a matrix

Using the method of elementary modifications of a matrix, the set of equations

$$\begin{aligned} 1\,a + 0\,b &= a \\ 0\,a + 1\,b &= b \end{aligned}$$

is modified with the help of elementary matrix modifications by sequential reducing of the degree of the polynomials into the form

$$\begin{aligned} pa + qb &= d \\ ra + sb &= 0 \end{aligned}$$

The source text of the corresponding m-file for solution of a Diophantine equation by means of elementary matrix modifications is given below.

```
function [d,p,q,r,s] = elmodmat(a,b,c)
% Solution of Diophantine equation ax + by = c
% by elementary matrix modifications:
% [ 1 0 a ]
% [ 0 1 b ]
% The aim is to modify the matrix by means of
% elementary operations into the form:
% [ p q d ]
% [ r s 0 ]
% Function uses Symbolic Math Toolbox
%
% test of equation correctness:
if isempty(find(a)) | isempty(find(b))
    display('Diophantine equation is not given correctly!')
    p=0; q=0; d=0; s=0; r=0;
    return
end
```

```
syms R X ; % symbolic variable definition
A=poly2sym(a,X); % symbolic polynomial a
B=poly2sym(b,X); % symbolic polynomial b
R1=[1,0,A ]; % the first row of the matrix
R2=[0,1,B ]; % the second row of the matrix
% test of founded divisor:
while norm(sym2poly(R2(3)),inf) > 100*eps
    aa=sym2poly(R1(3)); bb=sym2poly(R2(3));
    % normalization of the polynomials and matrix rows:
    R1=R1/aa(1); R2=R2/bb(1);
    aa=aa/aa(1); bb=bb/bb(1);
    % n is difference between the polynomial degrees:
    n=length(bb)-length(aa);
    if n < 0 % exchange of the matrix rows
        R=R1; R1=R2; R2=R;
        pom=aa; aa=bb; bb=pom;
        n=-n;
    end
    % reduction of the polynomial degree:
    R2=collect(R2-bb(1)/aa(1)*X^n*R1,X);
end
% the largest common divisor:
d=sym2poly(R1(3));
% particular solution:
p=sym2poly(R1(1)); q=sym2poly(R1(2));
% general solution of reduced equation:
r=sym2poly(R2(1)); s=sym2poly(R2(2));
return
```

4.3 Method of Indefinite Coefficients

The method of indefinite coefficients search for a particular solution of a Diophantine equation only. The degrees of the polynomials x and y have to be be guessed. After substitution of the polynomials with indefinite coefficients, values of the coefficients are computed by solving of a set of linear simultaneous equations. If there is no solution, it is necessary to modify the guesses of polynomial degrees. Computations are carried out using the Symbolic Toolbox, the corresponding function can be formulated in the following way.

```
function [d,p,q,r,s] = indefcoef(a,b,c)
% Solution of Diophantine equation ax + by = c
% by means of polynomials with indefinite coefficients
% polynomials p, q are searched as the minimum solution of equation ap + bq = d
% polynomial d is the greatest common divisor of polynomials a, b
% the greatest common divisor is computed with help of Maple function gcd
% r = -b/d, s = a/d
% function uses Symbolic Math Toolbox
%
% test of equation correctness
if isempty(find(a)) | isempty(find(b))
    display('Diophantine equation is given incorrectly!')
    p=0; q=0; d=0; s=0; r=0;
    return
end
syms X ; % definition of a symbolic variable
```

```
A=poly2sym(a,X); % symbolic polynomial a
B=poly2sym(b,X); % symbolic polynomial b
C=poly2sym(c,X); % symbolic polynomial c
% test if equation can be solved
D=maple('gcd',A,B);
if D  = maple('gcd',C,D)
    display('Diophantine equation has no solution!');
    return
end
d=sym2poly(D); dn=length(a)-length(b);
% first possible values of polynomial degrees
if dn >= 0
    m=0; n=dn;
else
    m=-dn; n=0;
end
error=1000*eps;
% test if polynomials were successfully found
while error>=1000*eps
    syms x y;
    % definition of symbolic indefinite coefficients
    for i=0:m
        xx(i+1)=sym(strcat('x',int2str(i)));
    end
    x=xx(1);
    for i=1:m
        x=x+xx(i+1)*X^i;
    end
    for i=0:n
        yy(i+1)=sym(strcat('y',int2str(i)));
    end
    y=yy(1);
    for i=1:n
        y=y+yy(i+1)*X^i;
    end
    % creation of symbolic Diophantine equation, determination of coefficients
    diofantine=collect(A*x+B*y-D,X);
    for i=1:length(a)+m
        eqn(i)=subs(diff(diofantine,X,i-1)/prod(1:i-1),X,0);
    end
    xy=xx; xy(m+2:m+n+2)=yy;
    % transformation of equations with coefficients into auxiliary polynomials
    eqnt=subs(eqn,xy(1),'t');
    for i=2:m+n+2
        eqnt=subs(eqnt,xy(i),strcat('t^',int2str(i)));
    end
    % numerical matrix of equation set
    for i=1:m+n+2
        pol=sym2poly(eqnt(i));
        AA(i,m+n+4-length(pol):m+n+3)=pol;
    end
    % solution of linear equation set
    xn=AA(:,1:m+n+2)\(-AA(:,m+n+3));
```

```
    % creation polynomials p,q
    for i=1:n+1
        q(i)=xn(i);
    end
    for i=n+2:m+n+2
        p(i-n-1)=xn(i);
    end
    % test of correct choice of polynomials
    error=norm(sym2poly(poly2sym(conv(a,p)+conv(b,q))-poly2sym(d)));
    if error < 1000*eps
        break; % polynomial were computed successfully
    else
        % correction of guessed polynomial degrees
        AA=zeros(size(AA)); n=n+1; m=m+1;
    end
end
% general solution of reduced equation
r=-deconv(b,d); s=deconv(a,d)
return
```

4.4 A Short Comparison of the Methods

The extended Euclidean algorithm is a fast method with the minimum number of steps. It needs the standard MATLAB only. The method of elementary matrix modifications are more time–consuming. To gain the minimum solution, it is necessary to reduce the degree of the polynomials systematically by removing of the leading coefficient. At seeming simplification of the procedure by removing another coefficient than the leading one, the solution may be a non–minimal one. It demands usage of the Symbolic Toolbox. Indefinite coefficient method lays high claims on a user, computations have sometimes to be repeated until the right degrees of the polynomials x and y are set up.

4.5 A Simple Example

To synthetize a discrete controller, the following Diophantine equation is needed to be resolved

$$\left(1 - z^{-1}\right) x + z^{-1} \left(1 - 2z^{-1}\right) y = 1$$

All the functions, `euklidalg` (extended Euclidean algorithm), `elmodmat` (elementary modifications of a matrix) and `indefcoef` (method of indefinite coefficients), give the same results:

$$\begin{array}{lll} d = 1 & p = 2z^{-1} + 1 & q = -1 \\ & r = -z^{-1}\left(1 - 2z^{-1}\right) & s = 1 - z^{-1} \end{array}$$

Then, the general solution of the equation is

$$\begin{aligned} x &= \left(2z^{-1} + 1\right) - z^{-1}\left(1 - 2z^{-1}\right) t \\ y &= -1 + \left(1 - z^{-1}\right) t \end{aligned}$$

Choice of the polynomial t as $t = 0$ gives the particular solution realizing the minimum controller:

$$x = 2z^{-1} + 1 \qquad y = -1$$

5 Conclusions

Solution of Diophantine equations is an essential task at synthesis of discrete controllers by means of algebraic control theory. The three simple methods of their solution with assistance of the MATLAB and its Symbolic Math toolboxes are described. Algorithms of all the methods were coded as MATLAB functions. These fuctions gave the same or very similar results depending on accuracy of numerical computations.

Acknowledgments

The work has been supported by the program No. MSM 223400007 of the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] *Using MATLAB.* The MathWorks, Inc., Natick, MA, 1998.

[2] V. Hanta. Solution to the Time-Optimal Discrete-Control Problem by Means of MATLAB. In *13th International Conference on Process Control PC '01*, pages P049/1–P049/5, Tatranské Matliare, 2001. Slovak University of Technology.

[3] V. Kučera. *Discrete Linear Control: the Polynomial Equation Approach.* Academia, Prague, 1979.

[4] S. MacLane and G. Birkhoff. *Algebra.* MacMillan Co., New York, 1988.

[5] C. Moler and P.J. Costa. *Symbolic Math Toolbox for Use with MATLAB.* The MathWorks, Inc., Natick, MA, 1997.

Ing. Vladimír Hanta, CSc.
Institute of Chemical Technology, Prague
Department of Computing and Control Engineering
Technická 1905, 166 28 Prague 6
phone: 00420-224 354 212, fax: 00420-224 355 053, e-mail: hantav@vscht.cz