Linear equations over the ring of stable transfer functions

Author: Jiří Lidinský
Czech Technical University in Prague
Faculty of Electrical Engineering

Supervisor: Martin Hromčík
Czech Technical University in Prague
Faculty of Electrical Engineering
Centre for Applied Cybernetics

January 18, 2004

Contents

Chapter 1

Introduction

Speaking in broad terms, we can distinguish three main approaches to analysis and design of linear control systems.

- The classical frequency-domain methods have evolved from the analysis of frequency responses of linear dynamics systems. Their main formal mathematical tool is the theory of functions of complex variable, in particular the Laplace transform in case of continuous-time and the Z-transform for the discrete-time systems. Systems are described in terms of their transfer functions reflecting just the external input-output relations, which brings about some difficulties related to the internal stability of the closed loop and to the realization of the compensator. The used formalism also causes that the domain of classical methods is reduced to SISO time-invariant linear systems. But despite these limitations, the classical methods still remain very popular, namely in the community of practicing engineers, for their simplicity and effectivity in many control problems encountered in industry.

  The classical methods are suitable for computational processing and a lot of software tools supporting the design process are available.

- The drawbacks of the classical approach and the increasing complexity of systems to be controlled resulted in new methods of synthesis, usually called the state-space or modern approach. The methods rely upon the exact definition of the state that is systematically used both for the deeper analysis of the plant (the state provides the insight into the internal structure of the system) and for the synthesis of the compensator (the knowledge of the state is employed for compensation). The main formal tools are differential equations, vector spaces and matrix theory. The modern methods are applicable to much wider class of situations than the classical ones, e.g. to MIMO and time-varying systems. However, they have not become so popular, namely among the practicing engineers, for the necessity of finding the state-space model and for the need of state reconstruction in case it cannot be directly measured.

From the numerical point of view, the state-space design methods for linear systems rely on numerical linear algebra which is a powerful tool. Since the 50s a lot of effort has been devoted to the development of accurate and numerically stable algorithms for linear algebra problems encountered in a large number of scientific computations.

- The origin of the polynomial or algebraic approach is dated to the early 70s. The polynomial matrices forming polynomial matrix fractions are introduced to handle MIMO cases. Systems are described by input-output relations, however the transfer functions are not regarded as functions of complex variable but as algebraic objects. The design procedure is then reduced to algebraic operations with rational and polynomial matrices, typically to solving algebraic Diophantine equations. This approach not only enables to resolve many existing control problems in a more elegant and unifying way but also provides further insight into the structure of the control systems and shows new relationships between various control tasks.

  The algebraic methods are closely related to the Czech science. Among many others, let us remember prof. Vladimír Kučera who is well known as the pioneer of algebraic approach in the field of control theory [8, 9, 11].

The main idea of the algebraic approach is that it gives a unified theoretic framework for various control tasks. The appropriate algebraic object must be chosen according to required property. The stability and properness (or causality) are the natural requirements. While it is advantageous to take polynomials and polynomial matrices for discrete-time systems, for continuous-time systems the stable proper rational functions or matrices must be considered.

A lot of work has been done in the field of polynomial algorithms in the recent years, however only few algorithms for stable rational functions can be found in the literature. These algorithms are not very numerically reliable and are mostly limited to SISO arguments. Two main approaches may be found. Algorithms that handle the arguments in the form of ratio of two polynomials or polynomial fraction takes advantage of transfer to polynomial problem. On the other side the algorithms that transform their arguments into the state-space description can be found. Mostly there are algorithms for stable coprime factorization. For instance in [13] the doubly coprime factorization is presented. It considers knowledge of stabilizing state-feedback and full-state observer. In [1] it is shown that computing the stable coprime factorization is equivalent to computing state-feedback and full-state observer. In [17] the numerically reliable algorithm for designing a stable coprime factorization is introduced.

Chapter 2

Objectives

The objective of this work is to develop new algorithms for solving algebraic equations over the ring of stable rational functions and matrices, implement them and study numerical properties of the new and existing methods.

In section 4.1 the basic algebraic definitions and facts are introduced. Moreover the role of algebra in the feedback control synthesis is mentioned. In sections 4.3 and 4.4 the existing algorithms are presented for computing the Bezout identity based on transformation to polynomial problem. The numerical properties are tested and are discussed there. In section 4.5 a new algorithm based on the state-space description of the system follows. The numerical properties are also discussed. Finally, we deal with doubly coprime factorization in the chapter 5. The existing routines and our new method is described here.

Chapter 3

Software framework

As a software framework for implementation and for testing of developed algorithms the pre-release version 3.0 of the Polynomial Toolbox for Matlab was taken, provided by the Polyx, Ltd. company [14]. There were several reasons for this decision. The Polynomial Toolbox (PT) offers objects and functions for easy manipulation with polynomials and polynomial matrices. It offers standard functions for handling polynomials such as addition, determinant, computing divisor, solving algebraic equation, etc. The PT is moreover focused on control analysis and synthesis problems. There is connection to Control systems toolbox for Matlab that was used in algorithms that handle arguments in the state-space description.

A simple example of work with PT follows. Suppose two polynomial matrices C and D

Example 1

```
>> C=[1+z,1;1-z,0], D=[1-z,1;1 1]

C =

     1 + z     1
     1 - z     0

D =

     1 - z     1
     1         1
```

we want to know a greatest left divisor

```
>> gld(C,D)

ans =

     1     0
     0     1
```

thus C, D are coprime, then there exists other two polynomial matrices E and F that satisfy $CE + DF = I$

```
>> [E,F]=axbyc(C,D,eye(2))

E =

         0         0
    1.0000   -1.0000

F =

         0         0
         0    1.0000
```

□

The prototype version 3.0 of the PT offers objects for matrix fractions, namely left `ldf` and right `rdf` matrix fraction and basic operations for them. Unfortunately there are not algorithms for computing algebraic equations with these arguments. Thus our goal is correct this deficiency.

Now, let us try to demonstrate solving a simple example published in [15] with PT and our function `axby4`[1] to demonstrate the integration of the program developed under this thesis into the PT 3.0 framework.

Example 2 Suppose the system

$$G(s) = \frac{1}{s(s-1)(s+1)} \begin{bmatrix} s(s+1)^2 & s(s-1)^2 \\ s+1 & (s+1)(s-1) \end{bmatrix}.$$

Its stable coprime factorization reads:

$$\begin{aligned} N &= \frac{1}{(s+1)^2} \begin{bmatrix} s(s+1) & (s+1)(s-2) \\ 1 & s+2 \end{bmatrix}, \\ M &= \frac{1}{(s+1)^2} \begin{bmatrix} s(s-1) & 1-s \\ 0 & (s+1)^2 \end{bmatrix}. \end{aligned}$$

Enter these systems to PT:

```
>> N=[s*(s+1),(s+1)*(s-2);1,s+2]/(s+1)^2

N =

     s + s^2     -2 - s + s^2    /    1 + 2s + s^2     0
     1            2 + s          /    0                1 + 2s + s^2

>> M=[s*(s-1),1-s;0,(s+1)^2]/(s+1)^2
```

[1]see section 4.4 and chapter 6 for more details

```
M =

    -s + s^2     1 - s           /    1 + 2s + s^2     0
     0           1 + 2s + s^2   /     0                1 + 2s + s^2
```

Functions N, M are in the form of right matrix fraction. To solve Bezout identity $NX + MY = 1$ type:

```
>> [X,Y]=axby4(N,M)

X.numerator =

     24 + 25s - 4.2s^2    -7 - 6.5s + 2s^2
    -6.2 + 16s + 27s^2     2 - 5s - 8.5s^2

X.denominator =

     1 + 2s + s^2     0
     0                1 + 2s + s^2

Y.numerator =

    -21 - 27s - 22s^2     7.5 + 8.5s + 6.5s^2
    -11 - 27s             4 + 11s + s^2

Y.denominator =

     1 + 2s + s^2     0
     0                1 + 2s + s^2
```

The function `axby4` was developed under this diploma thesis, see section 4.5 and chapter 6. Functions X, Y are again of the class right matrix fraction. It is easy to convince that Bezout identity holds

```
>> N*X+M*Y

ans =

    1.0000         0
         0    1.0000
```

The tests of stability and of properness are to be performed

```
>> isstable(X) & isstable(Y)

ans =

     1
```

```
>> isproper(Y) & isproper(Y)

ans =

     1
```

□

In the previous example the coprime factorization of the system $G(s)$ had to be done and then the Bezout identity was solved. In this case it is better to solve coprime factorization together with Bezout identity. Next the doubly coprime factorization of the system $G(s)$ will be computed by function `dcf` that was also written under this diploma thesis, see section 5.3 and chapter 6.

Example 3 Enter the system $G(s)$

```
>> G=[s*(s+1)^2,s*(s-1)^2;s+1,(s+1)*(s-1)]/(s*(s-1)*(s+1))

G =

     s + 2s^2 + s^3     s - 2s^2 + s^3    /   -s + s^3     0
     1 + s             -1 + s^2          /     0          -s + s^3
```

Solve doubly coprime factorization

```
>> [N,M,X,Y,Nt,Mt,Xt,Yt]=dcf(G,'rdf');
```

Finally check the result

```
>> [Y, X; -Nt, Mt]*[M, -Xt; N, Yt]

ans =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000
```

and the stability and properness:

```
>> isstable(N) & isstable(M) & isstable(X) & isstable(Y)

ans =

     1

>> isstable(Nt) & isstable(Mt) & isstable(Xt) & isstable(Yt)

ans =
     1

>> isproper(N) & isproper(M) & isproper(X) & isproper(Y)

ans =

     1

>> isproper(Nt) & isproper(Mt) & isproper(Xt) & isproper(Yt)

ans =

     1
```

□

Chapter 4

Diophantine equations

4.1 Theory

The center of our interest in this work is the Diophantine equation, respectively its special case called Bezout identity, in the ring of stable proper rational functions or stable proper rational matrices. In this section we introduce a few basic facts and definitions from algebra that cohere with our subject. You can find more complete explanation of this problem and proofs of presented facts in [18].

The ring, in our case, is a set of stable rational functions $\mathbf{S}$ or a set of matrices $\mathbf{M}(\mathbf{S})$ with elements from $\mathbf{S}$. It can be easily proved that a set $\mathbf{S}$ is a commutative ring with identity, and is a domain[1]. A set $\mathbf{M}(\mathbf{S})$ is noncommutative ring with identity. A set of all rational functions with real coefficients will be denoted $\mathbf{F}$ and a set of all matrices with elements from $\mathbf{F}$ will be denoted $\mathbf{M}(\mathbf{F})$. The set $\mathbf{F}$ is a field[2]. It is clear that the ratio of any two elements $a, b \in \mathbf{S}$ with $b \neq 0$ belongs to $\mathbf{F}$. Similarly suppose $A, B \in \mathbf{M}(\mathbf{S})$ of appropriate dimensions with $|B| \neq 0$; the right fraction AB^{-1} belongs to $\mathbf{M}(\mathbf{F})$. Left fraction analogously.

Definition 1 *Suppose $a \in \mathbf{S}$; then $d \in \mathbf{S}$ is a* divisor *of a, and a is a* multiple *of d, if there exists a $c \in \mathbf{S}$ such that $a = cd$. Suppose $a, b \in \mathbf{S}$ then $d \in \mathbf{S}$ is a* greatest common divisor *of a, b if (i) d is a divisor of both a and b, and (ii) d is a multiple of every common divisor of a, b. Two functions $a, b \in \mathbf{S}$ are* coprime *if every greatest common divisor of a, b is a unit* [3] *in* $\mathbf{S}$. □

For noncommutative rings, such as $\mathbf{M}(\mathbf{S})$, we define: $D \in \mathbf{M}(\mathbf{S})$ is a *right* divisor of $A \in \mathbf{M}(\mathbf{S})$ if there is $C \in \mathbf{M}(\mathbf{S})$ such that $A = CD$. In that case

[1] product of every pair of nonzero elements is nonzero

[2] every nonzero element is a unit

[3] function that has an inverse in the ring

A is a *left* multiple of D. Definition of the left divisor and right multiple is analogous.

It is clear that a unit in $\mathbf{S}$ is a function that has only stable zeros (including infinity — relative degree must be zero). These functions are sometimes called *miniphase functions.* Two functions $a, b \in \mathbf{S}$ are coprime if and only if they do not have common unstable zeros and at least one of them has relative degree zero.

Example 4 Suppose the following functions:

$$f_1(s) = \frac{s+1}{(s+2)^2},\ f_2(s) = \frac{s-1}{s+1},\ f_3(s) = \frac{s-1}{(s+1)^2}.$$

We can say that only $f_1(s)$ and $f_2(s)$ are coprime. Functions $f_2(s)$ and $f_3(s)$ have common unstable zero at $+1$. Functions $f_1(s)$ and $f_3(s)$ have common zero at infinity. □

Lemma 1 *Suppose $a, b, d \in \mathbf{S}$; then d is a greatest common divisor of a, b if and only if there exist $x, y \in \mathbf{S}$ such that $ax + by = d$. Functions $a, b \in \mathbf{S}$ are coprime if and only if there exist $x, y \in \mathbf{S}$ such that $ax + by = 1$.* □

The equation $ax+by = d$ is called Diophantine and its special case $ax+by = 1$ is called Bezout identity.

If there exists one particular solution $\hat{x}, \hat{y}$ of the Bezout identity then there are infinitely many solutions of the form $x = \hat{x} + bw$, $y = \hat{y} - aw$, where $w \in \mathbf{S}$ is a parameter.

In noncommutative ring $\mathbf{M}(\mathbf{S})$ one must distinguish between *left* ($\widetilde{A}\widetilde{X} + \widetilde{B}\widetilde{Y} = I$) and *right* ($XA+YB = I$) Bezout identity. Moreover, to be able to express all solutions that are parameterized by a parameter $W \in \mathbf{M}(\mathbf{S})$, the following two equations must be also satisfied: $\widetilde{B}A-\widetilde{A}B = 0$, $X\widetilde{Y}-Y\widetilde{X} = 0$. These four equations are usually expressed as a block matrix equation:

Definition 2 (Doubly coprime factorization) *Let X, Y, A, B, $\widetilde{X}$, $\widetilde{Y}$, $\widetilde{A}$, $\widetilde{B} \in \mathbf{M}(\mathbf{S})$; the equation*

$$\begin{bmatrix} Y & X \\ -\widetilde{A} & \widetilde{B} \end{bmatrix} \begin{bmatrix} B & -\widetilde{X} \\ A & \widetilde{Y} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{4.1}$$

is called doubly coprime factorization. □

Suppose there exist $\hat{X}, \hat{Y}, \hat{\widetilde{X}}, \hat{\widetilde{Y}} \in \mathbf{M}(\mathbf{S})$ such that (4.1) holds, then there are infinitely many solutions of (4.1) of the form $X = \hat{X} + W\widetilde{B}$, $Y = \hat{Y} - W\widetilde{A}$, $\widetilde{X} = \hat{\widetilde{X}} + BW$, $\widetilde{Y} = \hat{\widetilde{Y}} - AW$, where $W \in \mathbf{M}(\mathbf{S})$ is a parameter. More about doubly coprime factorization you can find in chapter 5.

The last notion we need is coprime factorization:

Definition 3 *Suppose $p \in \mathbf{F}$. An ordered pair (n, m) where $n, m \in \mathbf{S}$ is a* coprime factorization *of p if (i) $m \neq 0$, (ii) $p = n/m$, (iii) n and m are coprime.*

For noncommutative rings: Suppose $P \in \mathbf{M}(\mathbf{F})$. An ordered pair (N, M) where $N, M \in \mathbf{M}(\mathbf{S})$ is a right-coprime factorization *of P if (i) M is square and $|M| \neq 0$, (ii) $P = NM^{-1}$, (iii) N and M are right-coprime.* □

The left-coprime factorization is defined analogously.

In the literature, the coprime factorization of the system $G \in \mathbf{F}$ is often realized in the following way. G can be expressed as a ratio of two polynomials $G = b/a$, thus the coprime factorization can be computed as

$$N = \frac{b}{(s+1)^n},\ M = \frac{a}{(s+1)^n},$$

where $n = \max(\deg a, \deg b)$. It is not, of course, the only suitable factorization. We experimented with various denominators to find the best one in numerical point of view. Results that were achieved are presented in section 4.4.

For multivariable systems the factorization is a little bit more complicated. Reasonable algorithm may be found for instance in [15].

4.2 Stabilization

A great application of the algebraic approach is in the control theory. The result of the fundamental importance is the ability to express *all* stabilizable controllers for a given plant. The process of synthesis of a controller, robust or optimal in certain sense, is then divided into two steps. (i) The set of all stabilizable controllers is computed. This set is parameterized by a free parameter W. (ii) We are looking for a parameter W such that the requirements on behavior of the closed loop are satisfied. For more see Kučera in [11, 10]. The concept of computing the set of all stabilizable controllers will be introduced in this section.

Consider the standard unit feedback loop shown in figure 4.1. Let (N, M) be a stable coprime factorization of the plant P and (X, Y) be a stable coprime factorization of the controller C, thus $N, M, X, Y \in \mathbf{S}$. The closed loop is internally BIBO[4] stable if and only if the four transfer functions

$$\begin{bmatrix} y \\ u \end{bmatrix} = \frac{1}{1+PC} \begin{bmatrix} PC & P \\ C & -PC \end{bmatrix} \begin{bmatrix} r \\ d \end{bmatrix} = \frac{1}{NX+MY} \begin{bmatrix} NX & NY \\ MX & -NX \end{bmatrix} \begin{bmatrix} r \\ d \end{bmatrix} \tag{4.2}$$

are stable. We can see that this is satisfied if and only if term $NX + MY$ is a unit in $\mathbf{S}$. Thus a stabilizing controller C exists, and all controllers that

[4]bounded input results in bounded output

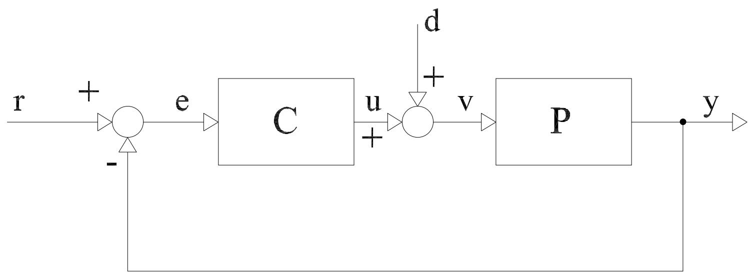

Figure 4.1: Closed loop

stabilize a given plant are generated by all solutions of the equation

$$NX + MY = 1. \tag{4.3}$$

So the set of controllers can be expressed in the form

$$C = \frac{\widehat{X} + MW}{\widehat{Y} - NW}, \ \widehat{Y} - NW \neq 0, \tag{4.4}$$

where the pair $\widehat{X}, \widehat{Y}$ is a particular solution of (4.3) and $W \in \mathbf{S}$ is a free parameter.

For multivariable systems it holds similarly. Only one must distinguish between the left and the right coprime factorization of the plant $P = NM^{-1} = \widetilde{M}^{-1}\widetilde{N}$, where $N, M, \widetilde{N}, \widetilde{M} \in \mathbf{M}(\mathbf{S})$. And the doubly coprime factorization (generalized Bezout identity) must be solved instead of Bezout identity (4.3). Thus the set of all controllers can be written down as

$$C = (\widehat{\widehat{X}} + MW)(\widehat{\widehat{Y}} - NW)^{-1} = (\widehat{Y} - W\widetilde{N})^{-1}(\widehat{X} + W\widetilde{M}), \tag{4.5}$$

where functions $\widehat{X}$, $\widehat{Y}$, $\widehat{\widehat{X}}$, $\widehat{\widehat{Y}} \in \mathbf{M}(\mathbf{S})$ satisfy doubly coprime factorization (4.1) and $W \in \mathbf{M}(\mathbf{S})$ is a free parameter.

In the following sections, numerical algorithms for solving Bezout identity will be studied. In addition to two existing approaches, our new method based on state-space description will be presented.

4.3 Substitution method

The following method converts the task — coprime factorization and Bezout identity solution — from the ring $\mathbf{S}$ to the ring of polynomials by appropriate substitution. Thus it benefits from the fact that algorithms for solving Diophantine equation in polynomials are relatively well known, see [5].

This method is popular in textbooks for its simplicity. For example in [3] the following procedure is presented:

Algorithm 1
Input: $G(s) \in \mathbf{F}$[5]
Output: $N(s), M(s), X(s), Y(s) \in \mathbf{S}$

1. Transform $G(s)$ to $\bar{G}(\lambda)$ under the mapping $s = (1-\lambda)/\lambda$. Write $\bar{G}(\lambda)$ as a ratio of coprime polynomials $\bar{G}(\lambda) = n(\lambda)/m(\lambda)$.
2. Solve Bezout identity $n(\lambda)x(\lambda) + m(\lambda)y(\lambda) = 1$ in the ring of polynomials.
3. Transform $n(\lambda)$, $m(\lambda)$, $x(\lambda)$, $y(\lambda)$ to $N(s)$, $M(s)$, $X(s)$, $Y(s)$ under the mapping $\lambda = 1/(s+1)$.

□

This simple method is applicable also for MIMO systems.

Vidyasagar, for example, uses bilinear transformation in [18]

$$z = \frac{s-\alpha}{s+\alpha},\ s = \alpha\frac{1+z}{1-z},\ \alpha > 0, \tag{4.6}$$

in the same way. This transformation takes the extended right half plane into the closed unit disc.

Although the above method is conceptually simple, it is not attractive in practice because changing variables requires much calculation. Another drawback is that the meaning of the obtained factorization is not clearly understood, according to [15].

4.4 Polynomial method

Algorithm for solving Bezout identity that will be presented in this chapter is based on transition from the ring of proper stable rational functions $\mathbf{S}$ to the ring of polynomials. Although this method is quite straightforward and not new, we derive it below, because its complete description in the literature is difficult to find. We also believe that our purely algebraic derivation of the X and Y denominators degrees is original. Moreover we studied numerical properties of this algorithm, see 4.4.1.

Suppose $N, M \in \mathbf{S}$ are given functions; we are looking for $X, Y \in \mathbf{S}$ such that $NX + MY = 1$. It is possible to write arguments N, M, X, Y as a ratio of two polynomials

$$NX + MY = \frac{n_n}{n_d}\frac{x_n}{x_d} + \frac{m_n}{m_d}\frac{y_n}{y_d} = 1. \tag{4.7}$$

[5]field of transfer functions

This equation can be rewritten as a polynomial equation with restriction conditions

$$\begin{aligned} n_n m_d x_n y_d + m_n n_d y_n x_d &= n_d m_d x_d y_d, \\ \deg x_n &\le \deg x_d, \\ \deg y_n &\le \deg y_d, \\ x_d, y_d &\in \text{set of stable polynomials.} \end{aligned} \tag{4.8}$$

We chose denominators of X and Y as identical polynomials $z = x_d = y_d$ to obtain a polynomial Diophantine equation. How to chose polynomial z is the crucial problem of this method. Following requirements on it must be satisfied:

1. *Stability of X, Y.* Polynomial z must be of course stable to satisfy third condition in equation (4.8).

2. *Solvability.* Assume that $N, M \in \mathbf{S}$ are coprime — requirement on solvability of the Bezout identity, see section 4.1. Although coprime functions on $\mathbf{S}$ cannot have common unstable zeros, common stable zeros are *not* excluded. It implies that it is possible to have polynomials n_n, m_n that are not coprime[6]. Let g be a greatest common divisor of polynomials n_n, m_n. Polynomial g must be a factor of the right hand side of the polynomial Diophantine equation (4.8) to guarantee solvability of this equation. Thus g must be a factor of z. We can do this because polynomial g is stable under initial assumption.

   If N, M are a product of stable coprime factorization of some $G \in \mathbf{F}$, then common factors can be canceled. If general arguments are allowed, the stability of g decides about coprimeness of N, M or in another words about solvability of Bezout identity.

3. *Properness of X, Y.* Polynomial Diophantine equation (4.8) will be solved by Sylvester matrix method. It will be shown here, that the condition on properness functions X, Y together with condition on solvability Sylvester matrix equation gives us a lower bound on the degree of polynomial z under the choice $x_d = y_d = z$.

   First, let us simplify notation of the equation (4.8)

   $$a x_n + b y_n = c z, \tag{4.9}$$

   where $a = \hat{n}_n \hat{m}_d$, $b = \hat{m}_n \hat{n}_d$, $c = \hat{n}_d \hat{m}_d h$, $gz = x_d = y_d$, $n_n = g\hat{n}_n$, $m_n = g\hat{m}_n$ $m_d = h\hat{m}_d$, $n_d = h\hat{n}_d$. Polynomial g is a greatest common divisor of n_n, m_n and polynomial h is a greatest common divisor of m_d,

[6] in the ring of polynomials

n_d. It means that common factors was canceled in the equation (4.9). We can rewrite equation (4.9) to the form of Sylvester matrix equation

$$\underbrace{\left[\begin{array}{ccccc|ccccc} a_0 & 0 & \cdots & 0 & 0 & b_0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & \cdots & 0 & 0 & b_1 & b_0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & a_{\delta a} & a_{\delta a-1} & 0 & 0 & \cdots & b_{\delta b} & b_{\delta b-1} \\ 0 & 0 & \cdots & 0 & a_{\delta a} & 0 & 0 & \cdots & 0 & b_{\delta b} \end{array}\right]}_{S} \left[\begin{array}{c} x_{n,0} \\ \vdots \\ x_{n,\delta x} \\ \hline y_{n,0} \\ \vdots \\ y_{n,\delta y} \end{array}\right] = \left[\begin{array}{c} cz_0 \\ \vdots \\ cz_{\delta cz} \end{array}\right], \tag{4.10}$$

where S is a column Sylvester matrix, that is composed of coefficients of polynomials a, b and $\delta a = \deg a$, $\delta b = \deg b$, ... Polynomials a and b are coprime, thus the Sylvester matrix S is nonsingular, see [7]. According to this, number of columns of S must be at least the same as number of rows to guarantee solvability of the equation (4.10). Precisely

$$\begin{aligned} \text{rows } (S) &= \deg c + \deg z + 1 \\ &= \deg a + \deg x + 1 = \deg b + \deg y + 1, \\ \text{columns } (S) &= \deg x + \deg y + 2, \\ \text{rows } (S) &\leq \text{columns } (S). \end{aligned} \tag{4.11}$$

From (4.11) and from condition on properness X, Y we obtain

$$\deg z \geq \deg a + \deg b - \deg c - 1, \tag{4.12}$$
$$\deg z \geq \deg x = \deg c + \deg z - \deg a, \tag{4.13}$$
$$\deg z \geq \deg y = \deg c + \deg z - \deg b. \tag{4.14}$$

Now two situations may appear: (i) functions N, M both have relative degree zero. Then $\deg a = \deg b = \deg c$. Degree of z can be chosen as $\deg z := \deg a - 1$ (the lowest value in the inequality (4.12)). It is obtained from inequalities (4.13),(4.14) that condition on properness of functions X, Y is satisfied under this choice, because $\deg x = \deg z$, $\deg y = \deg z$. (ii) Now assume that $\deg a < \deg c = \deg b$. It corresponds to situation that function N is strictly proper[7]. The choice $\deg z := \deg a - 1$ fails because

$$\deg x = \deg c + \deg z - \deg a < \deg z \;! \tag{4.15}$$

Thus X is not proper. The equation (4.15) must be relaxed: $\deg x = \deg c + \deg z - \deg a + \Delta$, where $\Delta > 0$. Situation where M is strictly

[7] $N(\infty) = 0$; only one of N, M may be strictly proper to be coprime

proper and $\deg b < \deg c = \deg a$ is similar. So the following estimations of degrees of unknown polynomials z, x_n, y_n are obtained:

$$\begin{aligned} \deg z &= \max(\deg a, \deg b) - 1, && (4.16) \\ \deg x_n &= \deg z, && (4.17) \\ \deg y_n &= \deg z. && (4.18) \end{aligned}$$

If we satisfy all of the previous given limitations, the polynomial z can be chosen arbitrarily. Finally the Sylvester matrix equation (4.10) is solved to obtain polynomial x_n, y_n.

Follows a summary of the method.

Algorithm 2
Input: $N, M \in \mathbf{S}$.
Output: $X, Y \in \mathbf{S}$, such that $NX + MY = 1$.

1. Compute a greatest common divisor $g = \gcd(n_n, m_n)$.
2. If g is not stable, then Bezout identity doesn't have solution.
3. Compute polynomials $a = \hat{n}_n m_d$, $b = \hat{m}_n n_d$, $c = n_d m_d$.
4. Let $\delta z = \max(\deg a, \deg b) - 1$, $\delta x = \delta z$, $\delta y = \delta z$.
5. Compute $z = (s+1)^{\delta z}$.
6. Compose and solve Sylvester matrix equation (4.10)
7. Finally compute $X = x_n/(zg)$, $Y = y_n/(zg)$.

□

Following two examples demonstrate the described algorithm.

Example 5 Suppose system $G(s) = 1/s$; we are looking for a set of all stabilizable controllers. Let us factorize the system as $G(s) = N/M$ where $N = 1/(s+1)$, $M = s/(s+1)$. Polynomial z may be chosen as $z = 1$. Equation (4.9) reads

$$x_n + s y_n = s + 1.$$

Solve the Sylvester equation (4.10)

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_{n,0} \\ y_{n,0} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The solution is $X = 1$, $Y = 1$. The set of all stabilizable controllers is

$$C = \frac{1 + \frac{s}{s+1} W}{1 - \frac{1}{s+1} W}, \quad W \in \mathbf{S}.$$

□

Example 6 Now let us have the functions

$$N(s) = \frac{s+3}{(s+1)(s^2+0.5s+1)},\ M(s) = \frac{s(s+3)}{s^2+0.5s+1}.$$

We are looking for functions $X, Y \in \mathbf{S}$ such that $NX + MY = 1$. Although given functions have common factor, for instance $U = (s+3)/(s+1)$, they are coprime because U is a unit[8] on $\mathbf{S}$. Greatest common divisor of n_n, m_n is $g = s+3$. Equation (4.9) is of the form

$$(s+3)x_n + s(s+3)(s+1)y_n = (s+1)(s^2+0.5s+1)z.$$

Degree of z must be at least 2 because

$$\delta z = \max(\deg a, \deg b) - 1 = \max(1,3) - 1 = 2,$$

and z has to be a multiple of $(s+3)$. Our choice is for instance $z = (s+3)^2$. Thus the polynomial Diophantine equation is obtained

$$(s+3)x_n + (s^3+4s^2+3s)y_n = s^5 + 7.5s^4 + 19.5s^3 + 23.5s^2 + 19.5s + 9.$$

Diophantine equation will be solved by Sylvester method

$$\left[\begin{array}{ccc|ccc} 3 & & & 0 & & \\ 1 & 3 & & 3 & 0 & \\ & 1 & 3 & 4 & 3 & 0 \\ & & 1 & 1 & 4 & 3 \\ & & & & 1 & 4 \\ & & & & & 1 \end{array}\right] \left[\begin{array}{c} x_0 \\ x_1 \\ x_2 \\ \hline y_0 \\ y_1 \\ y_2 \end{array}\right] = \left[\begin{array}{c} 9 \\ 19.5 \\ 23.5 \\ 19.5 \\ 7.5 \\ 1 \end{array}\right].$$

The result is:

$$X = \frac{-3s^2+3}{(s+3)^2},\ Y = \frac{s^2+3.5s+5.5}{(s+3)^3}.$$

Please note that the common factor of a and b need not be canceled which is beneficial from the numerical point of view. □

4.4.1 Numerical experiments

It was shown in section 4.2, that the first step in the process of controller synthesis is the choice of appropriate stable coprime factorization of a given system. In the previous section 4.4, we saw that during solving Bezout identity the appropriate stable denominator polynomial had to be chosen. The question is, what is an appropriate choice, how to place zeros of this polynomials. Should it depend on zeros and poles of a given system? The only condition, from the theoretical point of view, is that this zeros must

[8]has an inverse

lie in the stability area. In this section the results of numerical experiments will be presented. They demonstrate two things: numerical properties of the method described in the previous section, and the results of our effort to find appropriate polynomials that are to be chosen. Criterion is the numerical efficiency. It will be shown that the choice of appropriate stable polynomial has a big influence on numerical accuracy.

For discrete-time systems the region of stability is a unit disc. In this area, the one significant point can be found. This point is the origin of the complex plane. The polynomial that has multiple zero in origin has the form z^n, where n is the multiplicity. This polynomial is advantageous for computations. Unfortunately for continuous-time systems, where the region of stability is the open left half plane, such convenient point doesn't exist. In the literature the polynomial $(s + 1)^n$ is widely used for this purpose without a word of explanation.

Suppose a second-order polynomial $p(s) = s^2 + 2\zeta\omega_n s + \omega_n^2$, then ω_n is natural frequency and ζ is relative damping. In the following, four types of testing polynomials will be used: (P1) all zeros have the same natural frequency ω_n and uniformly varying relative damping ζ, (P2) zeros have the same ζ and varying ω_n, (P3) multiple real zeros, (P4) zeros are taken randomly. Figure 4.2 illustrates these four cases. If stable polynomial is required, the only stable part of this patterns will be taken.

First experiment: the dependence of numerical results on the degree of input arguments was tested. Tested systems N, M are generated as follows: numerators were random polynomials of degree δ, polynomials of type P4, and denominators were a polynomials of degree δ with natural frequency $\omega_n = 1$ and relative damping $1 \le \zeta \le \sqrt{2}/2$, thus polynomials of type P2. The Bezout identity $NX + MY = 1$ was solved for various δ. As a measure of numerical efficiency the norm $r = \|NX + MY - 1\|_\infty$ was taken. The results are presented in table 4.1.

δ	5	6	7	9	10	13
r	0	8.6e-11	8.9e-9	2.0e-8	3.8e-7	> 1

Table 4.1: Results of the first experiment

Second experiment: how numerical results depend on chosen stable factorization was studied. Tested systems $G(s) = n(s)/m(s)$ were generated as follows: Zeros of polynomial $n(s)$ had the same natural frequency $\omega_n = \alpha$, relative damping was from interval $\zeta \in \langle -1, 1)$, polynomial of type P1. Zeros of polynomial $m(s)$ had the same relative damping $\zeta = \sqrt{2}/2$, real parts of poles were from interval $\langle -\alpha, \alpha \rangle$, polynomial of type P2. Both of these

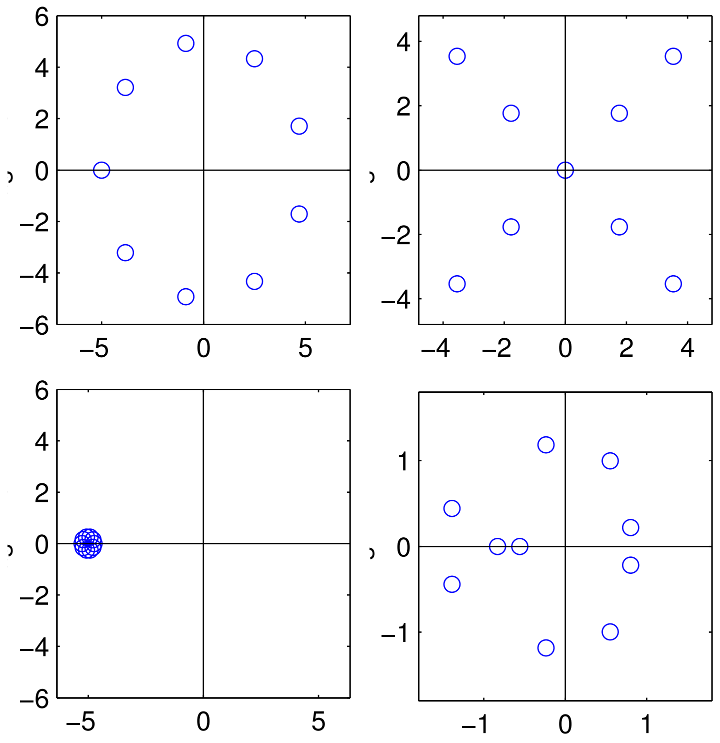

Figure 4.2: Patterns of testing polynomials P1–P4

polynomials were unstable. Tested system was factorized:

$$G(s) = \frac{N(s)}{M(s)},\ N(s) = \frac{n(s)}{z(s)},\ M(s) = \frac{m(s)}{z(s)},$$

where $z(s)$ is a *stable* polynomial of the type P1 with $\omega_n = \beta$ or P2 with $\zeta = \sqrt{2}/2$ and real parts of zeros $\in (-\beta, -0.1)$ or P3 with multiple zero in $-\beta$. Finally the Bezout identity $NX + MY = 1$ was solved for various α, β and for various types of $z(s)$. As the measure of numerical efficiency was the norm

$$r = \|NX + MY - 1\|_\infty. \tag{4.19}$$

The whole table of results, $(490 \times 4$ entries), is on the attached CD-ROM. The data presented in table 4.2 contains the maximum degree δ for each combination of α and β such that $r < 1e - 4$.

It is obvious, from the experiments, that this algorithm is efficient for systems $G(s)$ of degree less than seven, with poles and zeros in absolute value close to 1. Hence we can say that for this algorithm a quite appropriate choice is $(s+1)^n$ as commonly used in literature. Of course this leads to the Sylvester system of equations $Sx = b$ with S ill conditioned due to the tight

α	0.1	0.1	0.1	0.1	0.1	0.1	0.1
β	0.1	0.32	1	3.2	10	32	100
$\max_{z(s)\in P1}\delta$	6	10	6	4	2	2	2
$\max_{z(s)\in P2}\delta$	6	10	6	4	4	2	2
$\max_{z(s)\in P3}\delta$	6	10	6	4	2	2	2
α	0.32	0.32	0.32	0.32	0.32	0.32	0.32
β	0.1	0.32	1	3.2	10	32	100
$\max_{z(s)\in P1}\delta$	6	14	10	6	4	2	2
$\max_{z(s)\in P2}\delta$	8	10	12	8	4	4	4
$\max_{z(s)\in P3}\delta$	6	14	10	6	4	2	2
α	1	1	1	1	1	1	1
β	0.1	0.32	1	3.2	10	32	100
$\max_{z(s)\in P1}\delta$	6	12	16	8	4	4	2
$\max_{z(s)\in P2}\delta$	6	10	16	8	6	4	4
$\max_{z(s)\in P3}\delta$	6	12	14	6	4	2	2
α	3.2	3.2	3.2	3.2	3.2	3.2	3.2
β	0.1	0.32	1	3.2	10	32	100
$\max_{z(s)\in P1}\delta$	6	6	10	8	4	4	2
$\max_{z(s)\in P2}\delta$	6	6	10	8	6	4	4
$\max_{z(s)\in P3}\delta$	6	6	10	6	4	2	2
α	10	10	10	10	10	10	10
β	0.1	0.32	1	3.2	10	32	100
$\max_{z(s)\in P1}\delta$	2	6	6	6	4	4	2
$\max_{z(s)\in P2}\delta$	2	4	6	6	6	4	4
$\max_{z(s)\in P3}\delta$	2	6	6	6	4	2	2
α	32	32	32	32	32	32	32
β	0.1	0.32	1	3.2	10	32	100
$\max_{z(s)\in P1}\delta$	2	2	6	6	4	4	2
$\max_{z(s)\in P2}\delta$	2	2	4	6	6	4	4
$\max_{z(s)\in P3}\delta$	2	2	6	6	4	2	2
α	100	100	100	100	100	100	100
β	0.1	0.32	1	3.2	10	32	100
$\max_{z(s)\in P1}\delta$	2	2	2	6	4	4	2
$\max_{z(s)\in P2}\delta$	2	2	2	2	6	4	4
$\max_{z(s)\in P3}\delta$	2	2	2	6	4	2	2

Table 4.2: Results of the second experiment

alternating magnitudes of the combinator numbers in the $(s+1)^n$ expansion. Nevertheless, as experimentally proved, equally numerical difficulties appear for other alternative denominator patterns as well.

To evaluate the norm (4.19) the Matlab command

```
r = norm(NX+MY-1,inf)
```

was normally used. However, in certain cases (namely for higher degrees), the computation of $NX + MY - 1$ failed and we used the control systems toolbox function `nyquist` to evaluate estimation of this norm instead.

4.5 State-space based algorithm

Here we present a new algorithm for computing Bezout identity based on state-space description of the systems. The main idea is, that Bezout identity represents serial and parallel connection of four systems. The task is to find two of them, such that the overall system has transfer function equal to identity. This algorithm is applicable to MIMO systems without any restrictions.

Consider state-space representation of the systems in the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{4.20}$$

State matrices A, B, C, D will be marked by the index M, N, X, Y to be clear which system they stand for.

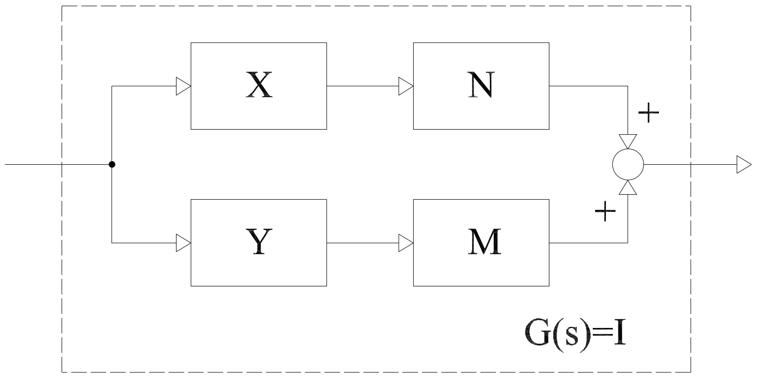

Figure 4.3: Schema that illustrates the main idea

Thus assume that equation $G(s) = N(s)X(s)+M(s)Y(s) = I$ represents serial and parallel connection of four systems. System matrices of the overall system $G(s)$ are composed of the connected system matrices in this way

$$
\begin{aligned}
A_I &= \begin{bmatrix} A_X & 0 & 0 & 0 \\ B_N C_X & A_N & 0 & 0 \\ 0 & 0 & A_Y & 0 \\ 0 & 0 & B_M C_Y & A_M \end{bmatrix}, && (4.21) \\
B_I &= \begin{bmatrix} B_X \\ B_N D_X \\ B_Y \\ B_M D_Y \end{bmatrix}, && (4.22) \\
C_I &= [\, D_N C_X \quad C_N \quad D_M C_Y \quad C_M \,], && (4.23) \\
D_I &= D_N D_X + D_M D_Y, && (4.24)
\end{aligned}
$$

and we require that this overall system has transfer function $G(s) = I$. It implies that overall system cannot have any dynamics. But we can see (4.21) that poles of $G(s)$ are given by the poles of the systems M, N, X, Y. Therefore we must find systems $X(s), Y(s) \in \mathbf{M}(\mathbf{S})$ such that dynamic of $G(s)$ is hidden. It means that poles of the system are unobservable and/or uncontrollable. So we are going to deal with controllability and observability of the $G(s)$ in the next paragraph.

Lemma 2 *[7] Suppose system $G(s)$ and its state-space description (4.20). Than the order of the controllable part of this system is*

$$n_1 = \mathit{rank}\, \mathcal{C},$$

the order of the observable part is

$$n_2 = \mathit{rank}\, \mathcal{O},$$

and the order of the controllable and *observable part of this system is*

$$n_3 = \mathit{rank}\, \mathcal{O}\mathcal{C},$$

where $\mathcal{C}$ is controllability matrix and $\mathcal{O}$ is observability matrix of the system $G(s)$. □

The overall system $G(s)$ cannot have any controllable and observable part, $n_3 = 0$, and therefore the product of the observability and controllability matrix must be a zero matrix $\mathcal{O}\mathcal{C} = 0$. Definition of the controllability and observability matrix is

$$\mathcal{O} = \begin{bmatrix} C_I \\ C_I A_I \\ C_I A_I^2 \\ \vdots \end{bmatrix}, \; \mathcal{C} = [\, B_I \quad A_I B_I \quad A_I^2 B_I \quad \cdots \,],$$

and the product $\mathcal{OC}$

$$\mathcal{OC} = \begin{bmatrix} C_I B_I & C_I A_I B_I & \cdots \\ C_I A_I B_I & C_I A_I^2 B_I & \cdots \\ C_I A_I^2 B_I & C_I A_I^3 B_I & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}.$$

Thus we obtained a system of matrix equations of the form

$$C_I A_I^i B_I = 0. \tag{4.25}$$

The power of the matrix A_I can be rewritten as

$$A_I^n = \begin{bmatrix} A_X^n & 0 & 0 & 0 \\ \sum_{i=0}^{n-1} A_N^i B_N C_X A_X^{n-1-i} & A_N^n & 0 & 0 \\ 0 & 0 & A_Y^n & 0 \\ 0 & 0 & \sum_{i=0}^{n-1} A_M^i B_M C_Y A_Y^{n-1-i} & A_M^n \end{bmatrix},$$

and the product $C_I A_I^n B_I$ reads

$$\begin{aligned} C_I A_I^n B_I &= D_N C_X A_X^n B_X + \\ &+ C_N \sum_{i=0}^{n-1} A_N^i B_N C_X A_X^{n-1-i} B_X + \\ &+ C_N A_N^n B_N D_X + D_M C_Y A_Y^n B_Y + \\ &+ C_M \sum_{i=0}^{n-1} A_M^i B_M C_Y A_Y^{n-1-i} B_Y + \\ &+ C_M A_M^n B_M D_Y. \end{aligned} \tag{4.26}$$

Let us present two simple examples that bring little light into equations we have presented above and indicate a further way.

Example 7 Suppose we have the following task: find all stabilizable controllers that stabilize the system which is composed of two serial connected integrators $G(s) = 1/s^2$. First we must factorize given system:

$$G(s) = \frac{1}{s^2} = \frac{\frac{1}{(s+1)^2}}{\frac{s^2}{(s+1)^2}} = \frac{N(s)}{M(s)}.$$

Second step, find a state-space representation of the systems $N(s), M(s)$. We choose this one:

$$A_N = \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix}, B_N = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C_N = [\,1 \quad 0\,], D_N = 0,$$

$$A_M = \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix}, B_M = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, C_M = [\,1/2 \quad -1\,], D_M = 1.$$

Now the system matrices A_X, A_Y must be chosen. It is the same operation as choosing denominators in previous algorithm, see section 4.4. System matrices must be stable and a minimum size must be at least one, see 4.5.1. Our choice is

$$A_X = -1, \; A_Y = -1.$$

From equation (4.24) we obtain

$$D_I = D_N D_X + D_M D_Y = 0D_X + 1D_Y = 1 \Rightarrow D_Y = 1.$$

Next the system of equations (4.26) will be solved

$$\begin{aligned} C_I B_I &= C_Y B_Y - 2 = 0 \Rightarrow C_Y B_Y = 2, \\ C_I A_I B_I &= -3C_Y B_Y + D_X + 3 = D_X - 3 = 0 \Rightarrow D_X = 3, \\ C_I A_I^2 B_I &= C_X B_X + 6C_Y B_Y - 2D_X - 4 = 0 \Rightarrow C_X B_X = -2. \end{aligned}$$

So we have matrix $D_X = 3$, and products $C_X B_X = -2, C_Y B_Y = 2$. But this is all we need to compose transfer functions of systems $X(s), Y(s)$

$$\begin{aligned} X(s) &= \frac{C_X B_X}{s+1} + D_X = \frac{3s+1}{s+1}, \\ Y(s) &= \frac{C_Y B_Y}{s+1} + D_Y = \frac{s+3}{s+1}. \end{aligned}$$

It is easy to verify that this is a solution of the Bezout identity. All stabilizable controllers can be obtained from (4.4). □

The whole calculation was very simple because the system matrices A_X, A_Y were scalar. Let us investigate a little bit complicated situation.

Example 8 The task will be changed to obtain systems $X(s)$ and $Y(s)$ of order two. For instance let us take

$$N(s) = \frac{1}{(s+1)(s+2)}, \; M(s) = \frac{s^2}{(s+1)^2}.$$

State-space description of the systems $N(s)$ and $M(s)$ is

$$A_N = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}, B_N = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, C_N = \begin{bmatrix} 2 & -2 \end{bmatrix}, D_N = 0,$$

$$A_M = \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix}, B_M = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, C_M = \begin{bmatrix} 1/2 & -1 \end{bmatrix}, D_M = 1.$$

Choose systems matrices A_X, A_Y

$$A_X = A_Y = \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix}.$$

From (4.24) we obtain

$$D_I = 0D_X + 1D_Y = 1 \Rightarrow D_Y = 1.$$

Next solve the system of matrix equations (4.26)

$$\begin{aligned}
C_I B_I &= C_Y B_Y - 2 = 0, \\
C_I A_I B_I &= D_X + C_Y A_Y B_Y - 2C_Y B_Y + 3 = \\
&= D_X - 3C_Y B_Y + C_Y \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} B_Y + 3 = 0, \\
C_I A_I^2 B_I &= C_X B_X - 3D_X + C_Y A_Y^2 B_Y - 2C_Y A_Y B_Y + 3C_Y B_Y - 4 = \\
&= C_X B_X - 3D_X + 6C_Y B_Y - 4C_Y \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} B_Y - 4 = 0, \\
C_I A_I^3 B_I &= -4C_X B_X + C_X \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} B_X + 7D_X - 10C_Y B_Y + \\
&+ 10C_Y \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} B_Y + 5 = 0, \\
C_I A_I^4 B_I &= 11C_X B_X - 5C_X \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} B_X - 15D_X + \\
&+ 15C_Y B_Y - 20C_Y \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} B_Y - 6 = 0.
\end{aligned}$$

We took advantage of the Jordan form of the matrices A_X a A_Y. Let matrix U be a square matrix with ones above the diagonal

$$U = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix},$$

and λ be an eigenvalue of the matrix A_X, we obtain:

$$A_X = \lambda I + U, \; A_X^2 = \lambda^2 I + 2\lambda U.$$

The result of the system equations is

$$\begin{aligned}
C_X B_X &= 1, \; C_X U B_X = -2, \; D_X = 3, \\
C_Y B_Y &= 2, \; C_Y U B_Y = 0.
\end{aligned}$$

This is, similarly to the previous example, all we need to compute the transfer functions $X(s)$, $Y(s)$.

$$\begin{aligned}
X(s) &= C_X (sI - A_X)^{-1} B_X + D_X = C_X \frac{\text{adj } (sI - A_X)}{\det(sI - A_X)} B_X + D_X = \\
&= \frac{C_X \text{adj} \begin{bmatrix} s+1 & 1 \\ 0 & s+1 \end{bmatrix} B_X}{(s+1)^2} + D_X =
\end{aligned}$$

$$\begin{aligned} &= \frac{(s+1)C_X B_X + C_X U B_X}{(s+1)^2} + 3 = \\ &= \frac{3s^2+7s+2}{(s+1)^2}, \\ Y(s) &= \frac{s^2+4s+3}{(s+1)^2}. \end{aligned}$$

It is easy to proof that these functions $X(s)$, $Y(s)$ satisfy the Bezout identity $N(s)X(s) + M(s)Y(s) = 1$. □

Let us try to generalize results from previous examples. We are looking for matrices $A_X, B_X, C_X, \ldots$ such that Bezout identity is satisfied. First step is the choice of system matrices A_X, A_Y. This step corresponds to the choice of denominator in the previous transfer-function method see section 4.4. The matrices must be chosen such that eigenvalues lie in the stability region. In the previous examples we can see that it was important to have the matrices in Jordan canonical form. We consider further that we can choose A_X, A_Y as a one Jordan block (it is not always possible, in the previous method we have seen that sometimes it is necessary to eliminate common stable zeros see section 4.4). Later we will show extension for more than one Jordan block.

It is known that power of a matrix in Jordan canonical form can be written out as a sum of matrices:

$$J^n = \sum_{i=0}^{k} \binom{n}{i} \lambda^{n-i} U^i, \tag{4.27}$$

where $k = \min(\delta_A - 1, n)$, λ is eigenvalue of J and U is a square matrix of the appropriate size with ones above the diagonal

$$U = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

System of equations (4.26) is generally nonlinear in unknowns B_X, C_X, D_X, B_Y, C_Y. However, powers of system matrices $A_X, A_X^2 \ldots$ can be rewritten according to (4.27) to obtain a system of linear equations in unknowns $B_X C_X$, $B_X U C_X$, $B_X U^2 C_X, \ldots$ that we are able to solve. Transfer functions of the resulting systems X, Y could be obtained from equation

$$\begin{aligned} X(s) &= C_X(sI - A_X)^{-1} B_X + D_X = \\ &= C_X \frac{\text{adj }(sI - A_X)}{\det(sI - A_X)} B_X + D_X = \\ &= \frac{\sum_{i=0}^{\delta-1} (s-\lambda)^{\delta-1-i} C_X U^i B_X}{\det(sI - A_X)} + D_X, \end{aligned} \tag{4.28}$$

where δ is a size of A_X. Transfer function of the system $Y(s)$ can be obtained similarly.

Now consider system matrices of X and Y in the multi-block Jordan form:

$$A_X = A_Y = \begin{bmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_m \end{bmatrix}, \quad B_{X,Y} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_m \end{bmatrix}, \tag{4.29}$$
$$C_{X,Y} = [\, C_1 \quad C_2 \quad \cdots \quad C_m \,],$$

where J_i is a Jordan block related to eigenvalue λ_i. The term $C_X A_X^n B_X$ falls apart

$$C_X A_X^n B_X = C_{X1} J_1^n B_{X1} + C_{X2} J_2^n B_{X2} + \cdots, \tag{4.30}$$

term $C_Y A_Y^n B_Y$ similarly. Thus the only extension of algorithm for system matrices A_X, A_Y build of more than one Jordan block lies in more unknowns.

The summary of the suggested algorithm follows in the form of a step-by-step procedure.

Algorithm 3
Input: matrices $N, M \in \mathbf{M}(\mathbf{S})$ of appropriate dimensions
Output: matrices $X, Y \in \mathbf{M}(\mathbf{S})$ such that $NX + MY = I$

1. Transform matrices N, M into state-space form.
2. Choose system matrices A_X, A_Y in Jordan canonical form, such that eigenvalues lie in the stable area. Denote the size of these matrices δ. Size of these matrices see 4.5.1.
3. Compose system of linear matrix equations (4.26) using (4.27), such that unknowns are $C_X B_X$, $C_X U B_X$, $C_X U^2 B_X, \ldots, C_X U^\delta B_X$, $C_Y B_Y$, $C_Y U B_Y$, $\ldots C_Y U^\delta B_Y$, D_X, D_Y.
4. Solve the system of equations
5. Compute transfer matrices of X, Y according to (4.28)

□

4.5.1 Degree estimation of system matrices A_X, A_Y

The minimal size of these matrices can be derived from the theory of state stabilization of the systems. For SISO systems we can imagine the system $G(s) = N/M$ that can be stabilized by a controller of order $n-1$, where n is the order of the system $G(s)$. This controller is composed from a reduced-order observer and a state-feedback. For MIMO system the reasoning is similar and the minimal order of the controller is $n - \operatorname{rank} C$, where C is output matrix of the state-space description of system $G(s)$, see [7]. Thus the minimal order of system matrices A_X and A_Y must be $n - \operatorname{rank} C$.

4.5.2 Numerical experiments

We will show numerical properties of the presented algorithm by a simple experiment. Tested system $G(s)$ was generated as a random system of order δ, with two inputs and two outputs by Matlab commands:

```
A = rand(d); B = rand(d,2);
C = rand(2,d); D = rand(2,2);
G = ss(A,B,C,D);
```

System was factorized as $G(s) = NM^{-1}$, where $N, M \in \mathbf{M}(\mathbf{S})$ are coprime. In Matlab:

```
[N,M,X,Y,Ntilde,Mtilde,Xtilde,Ytilde] = dcf(G);
```

where `dcf` is our doubly coprime factorization function, see section 5.3 for details. Finally the Bezout identity was solved by function `axby4` that is the implementation of this algorithm:

```
[X,Y] = axby4(Mtilde,Ntilde);
```

The results for orders $\delta = 2, \ldots, 13$ are presented in table 4.3. As a measure

δ	2	3	4	5	6	7
R	0	3.9e-15	2.4e-13	3.7e-14	1.5e-13	3.3e-10
δ	8	9	10	11	12	13
R	6.3e-08	8.1e-07	1.1e-07	1.7e-03	> 1	> 1

Table 4.3: Results of the numerical test

of numerical performance the infinity norm $R = \|NX + MY - I\|_\infty$ was chosen.

Although this method is original and can handle MIMO systems, it is obvious from the test that it is satisfactory usable for systems of order less than eleven. So this method does not significantly outperform the polynomial algorithm of section 4.4 in the SISO case, nevertheless, it works for MIMO systems in the contrast. As we will explain in this paragraph, this is not coincidence because the cause is similar. Let $Sx = b$ be a matrix expression of the systems of equations (4.26), where unknowns x are $C_X U^i B_X$ and $C_Y U^i B_Y$. The entries of matrix S are build from powers of stable matrices and it can be seen from equation (4.27), that we can express them as a sum of combination numbers. Consequence is, that the difference between the elements of matrix S may be huge and such matrix is typically ill conditioned. This situation illustrates graph 4.4 where the one typical matrix S is vizualized. Its condition number with respect to inversion is $5.17e + 20$.

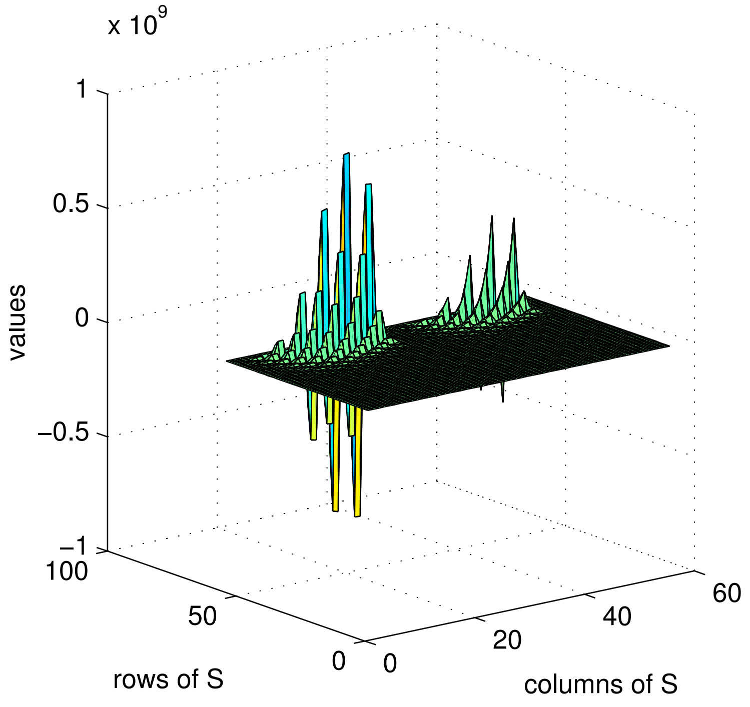

Figure 4.4: Typical structure of matrix S

Chapter 5

Doubly coprime factorization

We found, during the work on Bezout identity, that there is a related and in a sense more general task called Doubly coprime factorization (DCF) or sometimes generalized Bezout identity [13, 18]. This task connects coprime factorization of a given system together with Bezout identity in a block matrix equation. It seems to be more efficient and sensible to compute DCF than to compute coprime factorization and Bezout identity separately.

5.1 Theory

In the following theorem the existence conditions of DCF are presented.

Theorem 1 ([18] The existence of doubly coprime factorization) *Suppose system $G(s) \in \mathbf{M}(\mathbf{F})$, and let $G(s) = NM^{-1} = \tilde{M}^{-1}\tilde{N}$ be any right (respectively left) coprime factorization of $G(s)$. Suppose $X, Y \in \mathbf{M}(\mathbf{S})$ satisfy $XN + YM = I$. Then there exist $\tilde{X}, \tilde{Y} \in \mathbf{M}(\mathbf{S})$ such that*

$$\begin{bmatrix} Y & X \\ -\tilde{N} & \tilde{M} \end{bmatrix} \begin{bmatrix} M & -\tilde{X} \\ N & \tilde{Y} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{5.1}$$

The ordered pair of left hand side matrices in (5.1) is referred to as a doubly coprime factorization. □

Let us refer to a few notes

1. It is obvious that the block matrix equation (5.1) contains a left $G(s) = \tilde{M}^{-1}\tilde{N}$ (respectively right $G(s) = NM^{-1}$) coprime factorization of $G(S)$, a left $\tilde{N}\tilde{X} + \tilde{M}\tilde{Y} = I$ (respectively right $XN + YM = I$) Bezout identity and a left $C(s) = X^{-1}Y$ (respectively right $C(s) = \tilde{Y}\tilde{X}^{-1}$) coprime factorization of the stabilizable controller.

2. One must appreciate that DCF is more then four separate equations, because once $N, M, \tilde{N}, \tilde{M}$ have been selected, every X, Y such that $XN + YM = I$ determines a unique $\tilde{X}, \tilde{Y}$ such that (5.1) holds.

3. The existence of stable coprime factorization of the system $G(s) = \tilde{M}^{-1}\tilde{N} = NM^{-1}$ implies that the system $G(s)$ must be stabilizable and observable.

5.2 Existing algorithms

In [15] an algorithm is described for computing DCF that is based on the transfer matrix of a given system $G(s)$. The main idea of this algorithm is as follows. Suppose, we can solve generalized Bezout identity of a system $G(s)$ in the ring of polynomial matrices

$$\begin{bmatrix} Q & -\tilde{U} \\ P & \tilde{V} \end{bmatrix} \begin{bmatrix} V & U \\ -\tilde{P} & \tilde{Q} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix},$$

where $P, Q, U, V, \tilde{P}, \tilde{Q}, \tilde{U}, \tilde{V}$ are polynomial matrices and $G(s) = PQ^{-1} = \tilde{Q}^{-1}\tilde{P}$. Then DCF is obtained in four steps:

Algorithm 4
Input: system $G(s) \in \mathbf{M}(\mathbf{F})$
Output: systems $N, M, X, Y, \tilde{N}, \tilde{M}, \tilde{X}, \tilde{Y} \in \mathbf{M}(\mathbf{S})$

1. Find polynomial matrices Q_0 and $\tilde{Q}_0$ such that $N = PQ_0^{-1}, M = QQ_0^{-1}, \tilde{N} = \tilde{Q}_0^{-1}\tilde{P}, \tilde{D} = \tilde{Q}_0^{-1}\tilde{Q}$ gives a proper stable factorization of $G(s)$.
2. Compute $N, M, \tilde{N}, \tilde{M}$.
3. Compute polynomial matrices R and E such that
$$(s+1)^\tau Q_0 U = R(s+1)^\tau \tilde{D} + E,$$
where τ is max. degree in $\tilde{Q}$. Let $X = (s+1)^{-\tau}E$
4. Compute
$$Y = Q_0 V + R\tilde{N},\ \tilde{Y} = \tilde{V}\tilde{Q}_0 + NR,\ \tilde{X} = \tilde{U}\tilde{Q}_0 - DR.$$

□

Algorithms for computations with polynomial matrices are required. Namely the generalized Bezout identity solver and a procedure for division of two polynomial matrices.

In contrast, another factorization method presented in [13, 18] is fully based on state-space description and on state-space methods. We have a system $G(s)$ with the state-space description:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{5.2}$$

The following theorem shows connection between stabilizable state-feedback, full-observer and DCF.

Theorem 2 *[18] Given system (5.2), suppose the pairs (A, B), (A, C) are stabilizable and detectable. Select constant matrices F and K such that the matrices $A_c = A - BF$, $A_o = A - KC$ are both Hurwitz*[1]*. Then $G = NM^{-1} = \tilde{M}^{-1}\tilde{N}$ and*

$$\begin{bmatrix} Y & X \\ -\tilde{N} & \tilde{M} \end{bmatrix} \begin{bmatrix} M & -\tilde{X} \\ N & \tilde{Y} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}, \tag{5.3}$$

where

$$\begin{aligned}
\tilde{N} &= C(sI - A_o)^{-1}(B - KD) + D, \\
\tilde{M} &= -C(sI - A_o)^{-1}K + I, \\
\tilde{X} &= F(sI - A_c)^{-1}K, \\
\tilde{Y} &= (C - DF)(sI - A_c)^{-1}K + I, \\
N &= (C - DF)(sI - A_c)^{-1}B + D, \\
M &= -F(sI - A_c)^{-1}B + I, \\
X &= F(sI - A_o)^{-1}K, \\
Y &= F(sI - A_o)^{-1}(B - KD) + I.
\end{aligned}$$

□

In the next section a numerically reliable algorithm for computing DCF will be developed that uses theorem 2.

We found that a function for computing DCF is implemented in software SciLab [6]. Name of this function is `dcf` and it is a component of the robust toolbox. It is also based on theorem 2 and uses pole-placement algorithm. The right coprime factorization $G(s) = NM^{-1}$ and the right Bezout identity $XN + YM = I$ is computed separately from the left coprime factorization and the left Bezout identity. It means that pole-placement algorithm must be executed four times. Therefore this procedure is less effective than ours, described in 5.3, because we compute pole-placement only twice. Moreover generally it is not possible to compute the left and the right part of DCF separately, see theorem 1. The equation (5.1) holds, in this case, only if $F = \tilde{F}$ and $K = \tilde{K}$, where F, K are stabilizable state-feedback matrices computed for the left factorization and $\tilde{F}, \tilde{K}$ for the right factorization.

5.3 A Schur method

In the theorem 2 it was shown how to compute DCF if we find stabilizable state-feedback matrices F and K. We will present, in this section, sequential pole-shifting algorithm using the real Schur form that solve this problem. This algorithm was presented by Varga for pole assignment problem in [16]

[1]A matrix is *Hurwitz* if all its eigenvalues have negative real parts

and for computation of coprime factorizations of rational matrices in [17]. The main idea is that a matrix in the Schur form is triangular with its eigenvalues on the diagonal. Moreover, it is possible to arrange the eigenvalues in an arbitrary sequence, see [4]. Numerical reliability is due to the fact that only orthogonal transformations are used.

We extended this pole-shifting algorithm for the DCF task. Our extension lies in the following. It is necessary to compute the state-feedback F and the full-observer matrix K in separate processes because during computation of the matrix K the eigenvalues must be ordered in reverse direction (contrary to the matrix F). As a consequence, at the end of the pole-shifting, we obtain two stable systems, expressed in different bases. So we have to keep using orthogonal transformations to be able to compose DCF finally.

Let us suppose a system $G(s)$ and its state-space representation (5.2). Consider the state matrix A in the ordered real Schur form

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}. \tag{5.4}$$

It means that A is a real block upper triangular matrix with 1×1 and 2×2 blocks on the diagonal. The 1×1 blocks contain the real eigenvalues of A and the 2×2 blocks contain the complex conjugate pair of eigenvalues of A. Ordered means that the block A_{22} is 1×1 or 2×2 and the eigenvalues don't lie in the stability region. Now we can find stabilizable state-feedback matrix F in the form $F = [0, F_2]$ such that the eigenvalues of the matrix $A_{22} + B_2F_2$ lie in the stable region. The matrix with closed loop, in addition, stay in the real Schur form.

$$A + BF = \begin{bmatrix} A_{11} & A_{12} + B_1F_2 \\ 0 & A_{22} + B_2F_2 \end{bmatrix}. \tag{5.5}$$

It is advantageous because we can find orthogonal matrix Q such that $Q(A + BF)Q^T$ is again in *ordered* real Schur form. Therefore it is possible to repeat previous steps until we have a stable state matrix.

The same process can be used to find a stabilizable full-observer matrix K. Consider state representation (5.2) of a system $G(s)$, where system matrix A is in the *ordered* real Schur form

$$\begin{aligned} A &= \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \\ C &= \begin{bmatrix} C_1 & C_2 \end{bmatrix}, \end{aligned} \tag{5.6}$$

however now the ordered means that eigenvalues of the block A_{11} (size 1×1 or 2×2) don't lie in the stable region. So we can find observer matrix in the form $K = [K_1^T, 0]^T$, such that the eigenvalues of the matrix $A_{11} + K_1C_1$ lie in the stable region. The system matrix with closed loop remain in the real Schur form.

$$A + KC = \begin{bmatrix} A_{11} + K_1C_1 & A_{12} + K_1C_2 \\ 0 & A_{22} \end{bmatrix}. \tag{5.7}$$

At the end of this process we obtain two stable systems

$$\left[\begin{array}{c|c} sI - U(A+BF)U^T & U^TB \\ \hline CU & D \end{array}\right] \text{ and } \left[\begin{array}{c|c} sI - V(A+KC)V^T & V^TB \\ \hline CV & D \end{array}\right],$$

where U and V are accumulated orthogonal similarity transformations. The problem is that these systems are expressed in different bases. Therefore we have to store similarity transformations used during the stabilization to be able to transform systems from one base to another. Finally we can apply theorem 2 to compose DCF. The ideas enlightened above give rise to the following procedure.

Algorithm 5
Input: system $G(s) \in \mathbf{M}(\mathbf{F})$
Output: systems $N, M, X, Y, \tilde{N}, \tilde{M}, \tilde{X}, \tilde{Y} \in \mathbf{M}(\mathbf{S})$

1. Compute the orthogonal matrix Q such, that QAQ^T is in the ordered real Schur form.

2. Find stabilizing state-space feedback F.

   (a) Set $Ac = QAQ^T$, $Bc = Q^TB$, $Cc = CQ$, $F = 0$, $W = I$, $q = 0$

   (b) If $q = n$ go to step 3

   (c) Let A_{22} be the last elementary block of the matrix Ac of size k ($k = 1$ or 2) and let B_2 be the last k rows of the matrix Bc. If $\|B_2\| < \epsilon$ (ϵ a given tolerance) then $q \leftarrow q + k$ and go to step 2b. (unstabilizable and uncontrollable eigenvalue).

   (d) If the eigenvalue of A_{22} is not stable than compute F_2 such that $A_{22} + B_2F_2$ is stable.

   (e) Compute $Ac \leftarrow Ac + Bc[0, F_2]$, $F \leftarrow F + [0, F_2]$.

   (f) Compute orthogonal matrix U to move next eigenvalue to the last diagonal position of matrix Ac. Next update $Ac \leftarrow UAcU^T$, $Bc \leftarrow U^TBc$, $Cc \leftarrow CcU$, $q \leftarrow q + k$, $W \leftarrow WU$. Go to step 2b.

3. Find stabilizing observer K.

   (a) Set $Ao = QAQ^T$, $Bo = Q^TB$, $Co = CQ$, $K = 0$, $q = 0$.

   (b) If $q = n$ go to step 4.

   (c) Let A_{11} be the first elementary block of matrix Ao of size k ($k = 1$ or 2) and C_1 be the first k columns of the matrix Co. If $\|C_1\| < \epsilon$ (ϵ a given tolerance) then $q \leftarrow q + k$ and go to step 3b. (Not observable eigenvalue).

   (d) If the eigenvalue of A_{11} is not stable, then compute K_1 such that the $A_{11} + K_1C_1$ is stable.

(e) Compute $Ao \leftarrow Ao + [K_1^T, 0]^T Co$, $K \leftarrow K + [K_1^T, 0]$.

(f) Compute orthogonal matrix V to move next eigenvalue on the first diagonal position of the matrix Ao. Update $Ao \leftarrow V Ao V^T$, $Bo \leftarrow V^T Bo$, $Co \leftarrow CoV$, $q \leftarrow q+k$, $W \leftarrow V^T W$. Go to step 3b.

4. Compute

$$
\begin{aligned}
N &= \left[\begin{array}{c|c} sI - Ac & Bc \\ \hline Cc + DF & D \end{array}\right], &
\tilde{N} &= \left[\begin{array}{c|c} sI - Ao & Bo + KD \\ \hline Co & D \end{array}\right], \\
M &= \left[\begin{array}{c|c} sI - Ac & Bc \\ \hline F & I \end{array}\right], &
\tilde{M} &= \left[\begin{array}{c|c} sI - Ao & K \\ \hline Co & I \end{array}\right], \\
X &= \left[\begin{array}{c|c} sI - Ao & K \\ \hline FW^T & O \end{array}\right], &
\tilde{X} &= \left[\begin{array}{c|c} sI - Ac & W^T K \\ \hline F & O \end{array}\right], \\
Y &= \left[\begin{array}{c|c} sI - Ao & Bo + KD \\ \hline -FW^T & I \end{array}\right], &
\tilde{Y} &= \left[\begin{array}{c|c} sI - Ac & -W^T K \\ \hline Cc + DF & I \end{array}\right].
\end{aligned}
$$

□

Some notes follow:

1. It is clear that an arbitrary region of stability can be chosen. It must be of course symmetrical with respect to real axis to preserve realness of the involved systems. For instance this method could be used for discrete systems where the region of stability is a unit disc.

2. Numerical reliability of this algorithm is due to the use of orthogonal transformations.

3. Next numerical advantage rises from the fact that only unstable poles are handled. Stable poles are left without any change.

A first few steps that cohere with ordering of the real Schur form will be explained in the following example

Example 9 Consider system[2]

$$G(s) = \frac{1}{s(s-1)(s+1)} \begin{bmatrix} s(s+1)^2 & s(s-1)^2 \\ s+1 & (s+1)(s-1) \end{bmatrix},$$

and its minimal state-space representation:

$$A = \begin{bmatrix} -1 & 0.4082 & 0.1543 \\ 0 & 0 & 0.378 \\ 0 & 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 0.1091 & 0.5455 \\ 0.2673 & -0.5345 \\ -0.7071 & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} -2.619 & 1.069 & -2.828 \\ 0 & -1.871 & -0.7071 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

[2]this system was used as example in [15]

We can see, that system matrix A is already in the real Schur form. Diagonal entries are poles of system $G(s)$: -1, 0, +1. Thus the system is unstable. At first, we are looking for a matrix F such that matrix $A + BF$ is stable. The lower right entry, $A_{22} = 1$, contains an unstable eigenvalue, thus the matrix A is in the *ordered* real Schur form. Now it is easy to stabilize it:

$$A_{22} + B_2 F_2 = 1 + [\,-0.7071 \quad 0\,] \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = -1.$$

The result is for example $F_2 = [\,2.828 \quad 0\,]^T$. The resultant eigenvalue, in case -1, must be chosen inside the stable region. So the state matrix is:

$$A \leftarrow A + BF = \begin{bmatrix} -1 & 0.4082 & 0.4629 \\ 0 & 0 & 1.134 \\ 0 & 0 & -1 \end{bmatrix},$$

and must be reordered. We must compute orthogonal matrix U to swap eigenvalues to the following position:

$$A \leftarrow UAU^T = \begin{bmatrix} -1 & 2.665e-15 & 0.6172 \\ 0 & -1 & -1.134 \\ 0 & 0 & 0 \end{bmatrix},$$

such U is

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.75 & 0.6614 \\ 0 & 0.6614 & 0.75 \end{bmatrix}.$$

Moreover all system matrices must be transform $B \leftarrow U^T B$, $C \leftarrow CU$, $W \leftarrow WU$. Now the unstable eigenvalue 0 is on the position A_{22} and it must be stabilize. The same process is to be performed to obtain a full-state observer matrix F, only the system matrix must be ordered in reverse direction. □

5.3.1 Implementation

A few notes about the implementation of the Schur algorithm. Algorithm was implemented under Matlab [12] and Control systems toolbox. We used two matrix functions that could have a fundamental influence on numerical results: a *standard* real Schur decomposition and function for swapping diagonal blocks in the real Schur form. For Schur decomposition we used a Matlab built-in function named `schur`. For swapping diagonal blocks, the algorithm presented in [2] was implemented.

Now let us consider, we are going to swap two blocks, A_{11} and A_{22}, in a matrix

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}.$$

In the presence of rounding errors, the biggest concern in this algorithm is solving the Sylvester equation:

$$A_{11}X - XA_{22} = \gamma A_{12}.$$

It could possibly be ill-conditioned if A_{11} and A_{22} are close eigenvalues. In the extreme case, if A_{11} and A_{22} have the same eigenvalues, the Sylvester equation is singular and the solution X may be infinite. The suitable scalar γ should prevent this problem.

So the care must be taken in the process of deciding which eigenvalues swap next. It is obvious that if eigenvalues for stabilization will be taken in the order from bottom-right to top-left, during the stabilization, and from top-left to bottom-right, during the full-observer computing, only the stable and unstable eigenvalues will be swapped in each step. As a consequence, the worse case comes for systems that have stable and unstable poles close to the bound of stability.

5.3.2 Numerical experiments

Eigenvalues close to stability bound test. To ensure that the numerical problem discussed in previous section was successfully solved we present this test. Let

$$A = \begin{bmatrix} 5 & 8.686e-1 & 4.165e-1 & 8.308e-1 & 3.578e-1 & 4.458e-1 \\ 0 & -5e-4 & 5.706e-1 & 1.764e-1 & 5.749e-1 & 7.142e-1 \\ 0 & 0 & -5e-4-\epsilon & 3.917e-1 & 8.285e-1 & 3.285e-1 \\ 0 & 0 & 0 & -5e-4+\epsilon & 6.478e-1 & 1.897e-1 \\ 0 & 0 & 0 & 0 & -5e-4 & 3.551e-1 \\ 0 & 0 & 0 & 0 & 0 & -5e-4-\epsilon \end{bmatrix}$$

be a system matrix of the tested system $G(s)$. Where $\epsilon = 2.2204e-16$. Other system matrices was taken randomly

```
B = rand(6,2);
C = rand(2,6);
D = zeros(2,2);
G = ss(A,B,C,D);
```

The doubly coprime factorization was computed by the Matlab command:

```
[N,M,X,Y,Nt,Mt,Xt,Yt] = dcf(G,eps,0.5e-3);
```

Stability bound was moved to $-5e-4$. The infinity norms of computed results are:

$$\begin{aligned} \|XN + YM - I\|_\infty &= 7.538172e-09, \\ \|\tilde{N}\tilde{X} + \tilde{M}\tilde{Y} - I\|_\infty &= 1.922672e-09, \end{aligned}$$

which are quite good and proves the reliability of the ordered Schur method.

Numerical efficiency test. We tried to find dependence of results accuracy on the degree of input system. Tested systems were generated randomly with two inputs and two outputs by Matlab commands:

```
A=rand(d); B=rand(d,2);
C=rand(2,d); D=rand(2,2);
S=ss(A,B,C,D);
```

where `d` is the required degree. The doubly coprime factorization was computed:

```
[N,M,X,Y,Nt,Mt,Xt,Yt]=dcf(S);
```

Finally the result norm was evaluated:

```
R_1=norm(Nt*Xt+Mt*Yt-eye(2),inf);
R_2=norm(X*N+Y*M-eye(2),inf);
```

You can see the results in table 5.1 for degrees $\delta = 1, 2, \dots, 30$.

δ	1	2	3	4	5	6
R_1	1.3e-16	2.6e-14	1.3e-15	3.0e-13	1.9e-13	7.2e-14
R_2	1.3e-16	1.9e-14	1.3e-15	9.9e-14	5.6e-14	4.1e-14
δ	7	8	9	10	11	12
R_1	4.0e-13	1.3e-13	1.0e-12	4.1e-11	1.3e-11	1.7e-11
R_2	8.7e-13	4.5e-13	4.1e-12	2.8e-11	5.7e-11	7.4e-11
δ	13	14	15	16	17	18
R_1	5.6e-12	5.4e-10	1.4e-10	1.0e-10	2.2e-09	1.7e-07
R_2	5.5e-10	4.8e-08	2.2e-10	1.1e-09	1.6e-09	9.8e-08
δ	19	20	21	22	23	24
R_1	2.1e-09	9.5e-08	4.8e-08	6.2e-09	1.2e-07	5.9e-06
R_2	2.6e-09	1.3e-05	9.5e-08	4.4e-08	5.8e-07	3.9e-07
δ	25	26	27	28	29	30
R_1	4.8e-07	4.7e-04	1.3e-04	4.9e-09	8.4e-06	9.1e-07
R_2	1.3e-04	2.0e-02	7.1e-03	3.9e-08	2.8e-06	9.4e-06

Table 5.1: Results of numerical efficiency test

We can see from the table that this method is numerically efficient even for high degrees of input system. It is obvious from comparison with algorithms that solve Bezout identity, that it is much more advantageous to solve doubly coprime factorization instead of computing stable coprime factorization plus Bezout identity, if there is such possibility.

Chapter 6

Implemented functions

In this chapter, the list of implemented functions will be presented. All functions were written under the Matlab 6.0 and require the Polynomial Toolbox 3.0 (PT) and Control Systems Toolbox. You can find them on the attached CD-ROM.

- `[X,Y]=axby3(A,B)`
  This function computes Bezout identity $AX + BY = 1$ over the ring **S**, thus only SISO systems are supported. Region stability is open left-half complex plane. It is implementation of algorithm described in section 4.4. Arguments A, B must be of a `ldf` or `rdf` class of PT or `lti` system of Control Systems toolbox.

- `[X,Y]=axby4(A,B)`
  Solves Bezout identity $AX + BY = I$ over the $\mathbf{M}(\mathbf{S})$. Region stability is also open left-half complex plane. Implemented algorithm is described in section 4.5. Arguments A, B must be of a `ldf` or `rdf` class of PT or `lti` system of Control Systems toolbox.

  An auxiliary function `test_axby4` is associated with `axby4`. It involves all problematic examples we have collected. It is served for testing functionality and for testing suitable tolerances.

- `[N,M,X,Y,Nt,Mt,Xt,Yt]=dcf(G,output_type,tol,stab_bound)`
  Solves a doubly coprime factorization

  $$\begin{bmatrix} Y & X \\ -Nt & Mt \end{bmatrix} \begin{bmatrix} M & -Xt \\ N & Yt \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix},$$

  where $G \in \mathbf{M}(\mathbf{F})$, all output arguments are from $\mathbf{M}(\mathbf{S})$. Used algorithm is described in section 5.3. Class of input argument must be `lti` from control systems toolbox or a polynomial fraction `ldf`, `rdf`. Region of stability is open left-half complex plane. Bound of stability could be moved by optional argument `stab_bound`, default is 0 (bound is imaginary axes). Optional argument `output_type` specify a class of output arguments and may be `'tf'`, `'ss'`, `'zpk'`, `'ldf'` or `'rdf'`, default is `'ss'`.

The purpose of the function `test_dcf` is the same as that of `test_axby4`, it tests some properties of function `dcf`.

Chapter 7

Conclusion

Numerical solvers for Diophantine equation over the ring of proper stable rational functions and proper stable rational matrices were studied in this thesis. The achieved results can be summarized as follows:

- First we considered a known algorithm for solving Bezout identity. This one handles only SISO arguments and takes advantage of easy transfer to polynomial problem. The numerical properties therefore intimately cohere with properties of used polynomial methods. We determined that the stable coprime factorization must be performed carefully if it is required, because it has a big influence on numerical stability. Suitable choices of denominators were considered and exposed to extensive numerical testing. See section 4.4.1 and tables in the files on the attached CD-ROM.

- Although there is potential possibility for extension to MIMO case for previous method, the limitation on SISO arguments taken us to developing a new algorithm based on state-space representation. The numerical stability was studied and is comparable with previous method. Unfortunately the method assumes coprimeness of input arguments in wide sense, because the coprimeness in MIMO case wasn't fully understood.

- Despite the fact that Bezout identity may be used for general arguments, if we deal with controller synthesis, the stable coprime factorization must be performed first. In this case it is better to compute doubly coprime factorization. In the chapter 5 we proposed a new algorithm, that uses real Schur form of the state matrix and orthogonal transformations. The good numerical properties were demonstrated by computer experiments.

Acknowledgement

The author would like to thank Zdeněk Hurák from the Czech Technical University, Centre for Applied Cybernetics, for most valuable discussions concerning namely the basics of algebraic design approach, the concept of doubly coprime factorization, and polynomial methods for its computation.

Bibliography

[1] P. J. Antsaklis. Proper stable transfer matrix fectorizations and internal system descriptions. *IEEE transactions on automatic control*, AC-31(7):634–638, 1986.

[2] Z. Bai and J. W. Demmel. On swapping diagonal blocks in real schur form. *Linear Algebra and its Applications*, 186:73–95, 1993.

[3] J. Doyle, B. Francis, and A. Tannenbaum. *Feedback Control Theory*. Macmillan Publishing, 1990.

[4] G. H. Golub and C. F. Loan. *Matrix Computations*. The Johns Hopkins Press Ltd. London, 1990.

[5] D. Henrion. *Reliable Algorithms for Polynomial Matrices*. PhD thesis, Institute of Information Theory and Automation, Prague, 1998.

[6] INRIA. *scilab* [online]. 2003 [cit 2004-01-05]. ⟨`http://scilabsoft.inria.fr/`⟩.

[7] T. Kailath. *Linear systems*. Prentice-Hall, 1980.

[8] V. Kučera. *Discrete Linear Control (The Polynomial Equation Approach)*. Academia Prague, 1979.

[9] V. Kučera. *Analysis and Design of Discrete Linear Control Systems*. Academia Prague, 1991.

[10] V. Kučera. The algebraic approach to control system design. In K. J. Hunt, editor, *Polynomial Methods in Optimal Control and Filtering*, pages 1–28. Peter Peregrinus Ltd., London, 1993.

[11] V. Kučera. Diophantine equations in control — a survey. *Automatica*, 29(6):1361–1375, 1993.

[12] The MathWorks. *matlab* [online]. 2003 [cit 2004-01-05]. ⟨`http://www.mathworks.com`⟩.

[13] C. N. Nett, C. A. Jacobson, and M. J. Balas. A connection beween state-space and doubly coprime fractional representations. *IEEE Transactions on Automatic Control*, AC-29(9):831–832, 1984.

[14] Ltd. Polyx. *polyx* [online]. 2003 [cit 2004-01-05]. ⟨`http://www.polyx.com`⟩.

[15] K. Sugimoto and Y. Yamamoto. A polynomial matrix method for computing stable rational doubly coprime factorizations. *Systems & Control Letters*, 14(3):267–273, 1990.

[16] A. Varga. A schur method for pole assignment. *IEEE transactions on automatic control*, AC-26(2):517–519, 1981.

[17] A. Varga. Computation of coprime factorizations of rational matrices. *Linear Algebra and its Applications*, pages 83–115, 1998.

[18] M. Vidyasagar. *Control Systems Synthesis A Factorization Approach.* The MIT Press, London, 1987.