Appendix A
Solving Linear Matrix Inequality (LMI) Problems

In this section, we present a brief introduction about linear matrix inequalities which have been used extensively to solve the FDI problems described in this book.

A.1 Introduction to LMI

LMIs are matrix inequalities which are linear or affine in a set of matrix variables. They are essentially convex constraints and therefore many optimization problems with convex objective functions and LMI constraints can easily be solved efficiently using many existing software. This method has been very popular among control engineers in recent years. This is because a wide variety of control problems can be formulated as LMI problems.

An LMI has the following form:

$$F(x) = F_0 + x_1F_1 + \cdots + x_nF_n = F_0 + \sum_{i=1}^{n} x_iF_i > 0 \tag{A.1}$$

where $x \in \mathscr{R}^m$ is the vector of decision variables and $F_0, F_1, \ldots, F_n$ are given constant symmetric real matrices, i.e., $F_i = F_i^T,\ i = 0, \ldots, m$. The inequality symbol in the equation means $F(x)$ is positive definite, i.e., $u^TF(x)u > 0$ for all nonzero $u \in \mathscr{R}^n$. This matrix inequality is linear in the variables xi.

As an example, the Lyapunov inequality

$$A^TP + PA < 0 \tag{A.2}$$

where $A \in \mathscr{R}^{n\times n}$ is given and $X = X^T$ is the decision variable that can be expressed in the form of LMI (A.1) as follows: Let $P_1, P_2, \ldots, P_m$ be a basis for the symmetric $n \times n$ matrices ($m = n(n + 1)/2$), then take $F_0 = 0$ and $F_i = -A^TP_i - P_iA$.

J. Zhang et al., *Robust Observer-Based Fault Diagnosis for Nonlinear Systems Using MATLAB®*, Advances in Industrial Control,
DOI 10.1007/978-3-319-32324-4

A.1.1 Tricks Used in LMIs

Although many problems in control can be formulated as LMI problems, some of these problems result in nonlinear matrix inequalities. There are certain tricks which can be used to transform these nonlinear inequalities into suitable LMI forms. Some of the tricks which are often used in control are described here with suitable examples.

1. Change of variables By defining new variables, it is sometimes possible to linearize nonlinear matrix inequalities.

Example A.1 Synthesis of state feedback controller

The objective is to determine a matrix $F \in \mathscr{R}^{m\times n}$ such that all the eigenvalues of the matrix $A + BF \in \mathscr{R}^{n\times n}$ lie in the open left-half of the complex plane. Using Lyapunov theory, it can be shown that this is equivalent to find a matrix F and a positive definite matrix $P \in \mathscr{R}^{n\times n}$ such that the following inequality holds:

$$(A + BF)^T P + P(A + BF) < 0, \tag{A.3}$$

or

$$A^T P + PA + F^T B^T P + PBF < 0. \tag{A.4}$$

Note that the terms with products of F and P are nonlinear or bilinear. Let us multiply either side of the above equation by $Q = P^{-1}$. This gives

$$QA^T + AQ + QF^T B^T + BFQ < 0. \tag{A.5}$$

This is a new matrix inequality in the variables $Q > 0$ and F. But it is still nonlinear. Let us define a second new variable $L = FQ$. This gives

$$QA^T + AQ + L^T B^T + BL < 0. \tag{A.6}$$

This gives an LMI feasibility problem in the new variables $Q > 0$ and $L \in \mathscr{R}^{m\times n}$. After solving this LMI, the feedback matrix F and Lyapunov variable P can be recovered from $F = LQ^{-1}$ and $P = Q^{-1}$. This shows that by making a change of variables, we can obtain an LMI from a nonlinear matrix inequality.

2. The Schur Complement Schur's formula is used in transforming nonlinear inequalities of convex type into LMI. This says that the LMI

$$\begin{bmatrix} Q(x) & S(x) \\ S(x)^T & R(x) \end{bmatrix} < 0, \tag{A.7}$$

where $Q(x) = Q(x)^T$, $R(x) = R(x)^T$ and $S(x)$ depends affinely on x, is equivalent to

$$R(x) < 0, \quad Q(x) - S(x)R(x)^{-1}S(x)^T < 0. \tag{A.8}$$

In other words, the set of nonlinear inequalities (A.8) can be transformed into the LMI (A.7).

Example A.2 Consider the following matrix inequality:

$$A^T P + PA + PBR^{-1}B^T P + Q < 0, \tag{A.9}$$

where $P = P^T > 0$ and $R > 0$, is equivalent to

$$\begin{bmatrix} A^T P + PA + Q & PB \\ B^T P & -R \end{bmatrix} < 0, \tag{A.10}$$

3. The S-Procedure This procedure is adopted when we want to combine several quadratic inequalities into one single inequality. In many problems of control engineering we would like to make sure that a single quadratic function of $x \in \mathscr{R}^m$ is such that

$$F_0(x) \leq 0, \quad F_0(x) := x^T A_0 x + 2b_0 x + c_0, \tag{A.11}$$

whenever certain other quadratic functions are positive semi-definite, i.e.,

$$F_i(x) \geq 0, \quad F_i(x) := x^T A_i x + 2b_0 x + c_0, \quad i \in (1, 2, \ldots, q). \tag{A.12}$$

Example To illustrate the S-procedure, consider the case $i = 1$ for simplicity. We need to guarantee that $F_0(x) \leq 0$ for all x such that $F_1(x) \geq 0$.

If there exists a positive (or zero) scalar τ such that

$$F_{aug}(x) := F_0(x) + \tau F_1(x) \leq 0 \forall x, \text{ s.t } F_1(x) \geq 0, \tag{A.13}$$

then our goal is achieved. This is because $F_{aug}(x) \leq 0$ implies that $F_0(x) \leq 0$ if $\tau F_1(x) \geq 0$, since $F_0(x) \leq F_{aug}(x)$ if $F_1(x) \geq 0$,

By extending this to q number of inequality constraints gives the following:

$$F_0(x) \leq 0 \text{ whenever } F_i(x) \geq 0 \text{ holds if } F_0(x) + \sum_{i=1}^{q} \tau_i F_i(x) \leq 0, \ \tau_i \geq 0. \tag{A.14}$$

A.1.2 Solving LMI Using MATLAB Toolbox

The LMI toolbox of MATLAB provides a set of useful functions to solve LMIs. Some of these functions are discussed here with sample codes.

Step-1: Initialization At the beginning, initialize the LMI description with the command *setlmis([])*. Note that this function does not take any parameter.

Step-2: Defining the Decision Variables Next, it is necessary to define the decision variables, i.e., the unknown variables of the LMI problem. As an example, consider the LMI $C^T XC < 0$, where C is a constant matrix and X is the matrix of decision variables. It is defined using the function *lmivar* which has the following syntax.

$$X = \text{lmivar(type, structure)}.$$

This command allows us to define several forms of decision matrices such as symmetrical matrices, rectangular matrices or matrices of other type. Depending on the selected matrix type, the structure contains different information. Thus, first we define the *type* and then define the *structure* which depends on the *type*.

- If type $= 1$, this implies that the matrix X is square and symmetrical. The structure element $(i, 1)$ specifies the dimension of the ith-block, while structure element $(i, 2)$ specifies the type of the ith-block (1 for full, 0 for scalar and -1 for zero block).
- If type $= 2$, matrix X is rectangular of size $m \times n$ as specified in *structure* $= [m, n]$.
- If type $= 3$, matrix X is of other type.

Step-3: Define the LMIs one by one This is done with the command *lmiterm*. The syntax of the command is

$$\text{lmiterm(termID, A, B, flag)}.$$

The *lmiterm* takes 3 or 4 arguments. The first argument *termID* is a 1×4 vector. The first element of this vector indicates which LMI is defined. The second and third entry in this vector gives the position of the term being defined. And the fourth entry indicates which LMI decision variable is involved. It can be 0, X or X depending on whether the term is constant, of type AXB or $AX^T B$. The second and third arguments of *lmiterm* function are the left and right multiplier of the decision matrix. If the *flag* is set to 's', it permits to specify with a single command that the given term and its symmetrical appears in the LMI.

Example A.3 Consider the following set of LMIs:

$$\begin{bmatrix} CX^T C^T + B^T YA & XF \\ F^T X & Y \end{bmatrix}, \tag{A.15}$$

$$DXD^T > 0. \tag{A.16}$$

Here we have two decision variables X and Y, and two LMIs. Let Y be a full symmetric matrix of dimension 5 and X has 5 blocks and the dimensions of the various block matrices are 5, 4, 3, 1 and 2.

First, we define the variables X and Y using *lmivar* function as follows:

```
structureX=[5,1;4,1;3,1;1,0;2,1];
X=lmivar(1,structureX);

structureY=[5,1];
Y=lmivar(1,structureY);
```

Then, we define the LMIs using *lmiterm* function as follows:

```
% LMI (A.15)
lmiterm([1 1 1 X],C,C');            % This represents term CXC'
lmiterm([1 1 1 Y],B',A);            % This represents the term B'
    YA
lmiterm([1 1 2 X],1,F);                 % This represents term XF
lmiterm([1 2 1 X],F',1);                % This represents term F'
    X
lmiterm([1 2 2 Y],1,1);                 % This represents term Y

% LMI (A.16)
lmiterm([-1,1,1,X],D,D');               % Term DXD > 0
```

Lastly, we create an LMI object using the following command.

```
myLMIsystem=getlmis;
```

A.2 Examples of Applying LMIs to Solve Various Control Problems

Example A.4 Determine the stability of linear time-invariant systems
Consider the linear time-invariant system

$$\dot{x} = Ax.$$

The system is stable provided the following inequalities are satisfied:

$$P > 0, \quad A^T P + PA < 0.$$

These two inequalities can be combined into a single LMI as

$$\begin{bmatrix} A^T P + PA & 0 \\ 0 & -P \end{bmatrix} < 0.$$

The MATLAB code for solving this stability problem for $A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$ is given as follows.

Programme example_A.4.m

```
A=[0 1;-2 -3];

% define unknown matrix which need to be determined by LMI
setlmis([])
P=lmivar(1,[size(A,1) 1]);  % Specify the structure and size of P
     .

% define LMI
lmiterm([1 1 1 P],1,A,'s');      % A'P+PA
lmiterm([1 1 2 0],1);            % 0
lmiterm([1 2 2 P],-1,1);         % P>0

LMISYS = getlmis;

[tmin,Psol]=feasp(LMISYS);
P=dec2mat(LMISYS,Psol,P)
```

After running this programme we get $t_{\min} = -2.615451$ and $P = \begin{bmatrix} 65.9992 & 12.8946 \\ 12.8946 & 15.1836 \end{bmatrix}$.

Example A.5 The LQR Problem: Solution of Riccati Equation.

Consider a system represented by the following linear continuous-time state equations:

$$\dot{x} = Ax + Bu, \quad x(0) = x_0,$$

where $x \in \mathscr{R}^n$ is the state vector, $u \in \mathscr{R}^m$ is the input vector, A and B are known matrices of appropriate dimensions. The objective is to determine the control input u which minimizes the following performance index:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt,$$

where $Q \in \mathscr{R}^{n \times n}$ is a real symmetric positive semi-definite matrix and $R \in \mathscr{R}^{p \times p}$ is a real positive definite matrix.

The optimal control input which minimizes J is given by

$$u(t) = R^{-1}B^T Px(t) = Kx(t), \quad K = R^{-1}B^T P, \tag{A.17}$$

where the matrix P is obtained by solving the following Riccati equation:

$$A^T P + PA + PBR^{-1}B^T P + Q < 0,\ P > 0,\ R > 0.$$

Note that the Riccati equation, in contrast to Lyapunov equations, is a nonlinear equation in P. This is because the quadratic term $PBR^{-1}B^T P$ appears in the inequality. Using the Schur Complement, we can represent this inequality as LMIs as follows:

$$\begin{gathered} P > 0,\ Q > 0,\ R > 0 \text{ and} \\ \begin{bmatrix} A^T P + PA + Q & PB \\ B^T P & -R \end{bmatrix} < 0. \end{gathered} \tag{A.18}$$

The MATLAB code for computing the gain K is given as follows:

Programme example_A.5.m

```
A=[-2 1;1 -1];
B=[1;1];

setlmis([])

% Specify the structure and size of P,Q and R
P=lmivar(1,[2,1]);
Q=lmivar(1,[2,1]);
R=lmivar(1,[1,1]);

% define LMIs
lmiterm([-1 1 1 P],1,1);        % P>0

lmiterm([-2 1 1 Q],1,1);        % Q>0

lmiterm([-3 1 1 R],1,1);        % R>0

% LMI (A.18)
lmiterm([4 1 1 P],A',1,'s');
lmiterm([4 1 1 Q],1,1);
lmiterm([4 1 2 P],1,B);

lmiterm([4 2 2 R],-1,1);

LMIs = getlmis;
```

```
[TMIN,XFEAS] = feasp(LMIs);

Q=dec2mat(LMIs,XFEAS,Q);
P=dec2mat(LMIs,XFEAS,P);
R=dec2mat(LMIs,XFEAS,R);

K=inv(R)*B'*P          % Compute control gain
```

After solving the LMIs, the control gain K is obtained as $K = [\,0.2454\ \ 0.2210\,]$.

Example A.6 H_∞ controller using full state feedback for the system

$$\dot{x} = Ax + B_1 w + B_2 u,$$

$$z = C_1 x + D_{11} w + D_{12} u.$$

The controller parameters can be obtained by solving the following LMIs:

$$Y > 0$$

$$\begin{bmatrix} YA^T + AY + Z^T B_2^T + B_2 Z & B_1 & YC_1^T + Z^T D_{12}^T \\ B_1^T & -\gamma I & D_{11}^T \\ C_1 Y + D_{12} Z & D_{11} & -\gamma I \end{bmatrix} < 0 \qquad \text{(A.19)}$$

The controller gain can be calculated from $F = ZY^{-1}$. For a predefined $\gamma = 1$, we have the following code:

Programme example_A.6.m

```
% known parameters
A=[0.8 -.25 ;1 0];
B1=[0 ; 0.1];
B2=[.1; 0.03];
C1=[0 .1];
D11=0.05;
D12=0.1;
gama=1;
setlmis([])
```

```
% Specify the structure and size of Y and Z
Y=lmivar(1,[2,1]);   % Symmetric Matrix 2 by 2
Z=lmivar(2,[1,2]);   % vector 1 by 2

lmiterm([-1 1 1 Y],1,1);   % Y>0

% LMI (A.19)
lmiterm([2 1 1 Y],A,1,'s');
lmiterm([2 1 1 Z],B2,1,'s');
lmiterm([2 1 2 0],B1);
lmiterm([2 1 3 Y],1,C1');
lmiterm([2 1 3 -Z],1,D12');
lmiterm([2 2 2 0],-gama);
lmiterm([2 2 3 0],D11');
lmiterm([2 3 3 0],-gama);

LMIs = getlmis;
options = [1e-5,0,0,0,0] ;
[TMIN,XFEAS] = feasp(LMIs,options,0);

Y=dec2mat(LMIs,XFEAS,Y);
Z=dec2mat(LMIs,XFEAS,Z);
F=Z*inv(Y)         % state feedback gain
```

By solving the MATLAB programme, we have $F = [\,-20.3758 \;\; -6.1154\,]$. It can be verified that $A + B_2F = \begin{bmatrix} -1.2376 & -0.8615 \\ 0.3887 & -0.1835 \end{bmatrix}$ and its eigenvalues are $-0.7105 + 0.2390i$ and $-0.7105 - 0.2390i$, which shows the system is stable.

Appendix B
YALMIP Toolbox: A Short Tutorial

Most of the MATLAB programs which are being used to compute the unknown parameters of the proposed observers are formulated using YALMIP toolbox. YALMIP **Y**et **A**nother **LMI** (linear matrix inequality) **P**arser is a modelling language for advanced modeling and solution of convex and nonconvex optimization problems. It is available freely to be used by researchers. For more information log on to the following:

http://users.isy.liu.se/johan1/yalmip/

It should be noted that YALMIP is only a modelling language and it requires some underlying solvers such as CPLEX, Sedumi, SDPT3, etc. So you should have one of these solvers in your computer.

In this section, some basic commands of YALMIP toolbox will be introduced [1].

B.1 Defining Decision Variables

The decision variables constitute the important component in any optimization problem. In YALMIP these variables are represented by the command *sdpvar*. A rectangular matrix **P** with **n** rows and **m** columns is represented by the command

```
P=sdpvar(n,m);
```

A square matrix $\mathbf{P} \in \mathscr{R}^{n\times n}$ is by default, a symmetric matrix and is defined either by the following command:

```
P=sdpvar(n,n,'symmetric','real');
```

or it is defined using only the dimension argument as

```
P=sdpvar(n,n);
```

We can include a third argument to represent different types of variables such as Toeplitz, Hankel, diagonal matrices using the following commands:

J. Zhang et al., *Robust Observer-Based Fault Diagnosis for Nonlinear Systems Using MATLAB®*, Advances in Industrial Control,
DOI 10.1007/978-3-319-32324-4

```
X=sdpvar(n,1);
D=diag(X);        % Diagonal matrix
H=hankel(X);      % Hankel matrix
T=toeplitz(X);    % Toeplitz matrix
```

B.2 Defining Constraints

The most commonly used constraints in YALMIP are element-wise, semi-definite, and equality constraints. For example, the code below generates a constraint to define matrices X to be positive definite and Y to be negative definite.

```
X=sdpvar(n,n);
Y=sdpvar(n,n);
constraint=[X > 0,Y < 0];
```

In addition to these standard constraints, YALMIP also supports definition of integrity constraints, second-order cone constraints, and sum of squares constraints.

B.3 Solving Optimization Problems

Once all variables and constraints have been defined, the optimization problem using the command `solvesdp` can be solved. For example, a linear program

$$\min c^T x \text{ subject to } Ax <= b$$

can be solved with the following piece of code:

```
x=sdpvar(length(c),1);
constraint=[A*x<=b];
h=c'*x;
solvesdp(constraint,h);
solution=double(x);
```

If we only look for a feasible solution, we can omit the objective function and use `solvesdp(constraint)`.

B.4 Solving Various Control Problems Using YALMIP

As discussed in Appendix A, we illustrate how to solve some of the control problems using YALMIP.

Example B.1 Determine the stability of Linear Time Invariant Systems

The system is stable provided the following inequalities are satisfied:

$$P > 0, \quad A^T P + PA < 0.$$

These two inequalities can be combined into a single LMI as:

$$\begin{bmatrix} A^T P + PA & 0 \\ 0 & -P \end{bmatrix} < 0.$$

The MATLAB code for solving this stability problem for $A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$ is given as follows.

Example_B.1.m

```
%_____________________________________________________
% Solution of Lyapunov Inequality A'P+PA<0, P>0
clear all
%_____________________________________________________
% System Matrix

A=[0 1
    -2 -3];
%_____________________________________________
% Define unknown variables

n=size(A,1);
P=sdpvar(n,n);

M1=[A'*P+P*A zeros(n,1)
    zeros(n,1) -P];

const=[M1<0];

solvesdp(const);

P=double(P)
```

Example B.2 Solving Linear Quadratic Regulator Problem

The LQR can be designed by solving the following Riccati equation:

$$A^T P + PA + PBR^{-1}B^T P + Q < 0,$$

where P, Q, R are symmetric positive definite matrices.

This inequality can be transformed into the LMIs as follows:

$$P > 0,\ Q > 0,\ R > 0 \text{ and}$$
$$\begin{bmatrix} A^T P + PA + Q & PB \\ B^T P & -R \end{bmatrix} < 0.$$

The MATLAB code for solving this LQR problem for $A = \begin{bmatrix} -2 & 1 \\ 1 & -1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ is given as follows.

Example_B.2.m

```
%_____________________________________________________
% Solution of Optimal Control Problem
clear all
%_____________________________________________________
% System matrices
A=[-2 1
    1 -1];
B=[1 1]';
%_____________________________________________
% Define unknown variables
n=size(A,1);
[n1,m]=size(B);

P=sdpvar(n,n);
Q=sdpvar(n,n);
R=sdpvar(m);

M1=[A'*P+P*A+Q P*B
    B'*P -R];

const=[M1<0,-P<0,-Q<0,-R<0];

solvesdp(const);

P=double(P)
Q=double(Q)
R=double(R)

K=inv(R)*B'*P
```

The controller gain K is obtained as $K = [0.2072\ 0.2182]$.

Example B.3 $\boldsymbol{H_\infty}$ State Feedback Control

Consider the discrete-time time-invariant linear system

$$\begin{aligned} x(k+1) &= Ax(k) + B_w w(k) + B_u u(k), \\ z(k) &= C_z x(k) + D_{zw} w(k) + D_{zu} u(k), \\ y(k) &= C_y x(k) + D_{yw} w(k), \end{aligned} \tag{B.1}$$

where the state vector $x \in R^n$ and all other matrices and vectors have appropriate dimensions. Assume that the state vector $x(k)$ is available for feedback and the state information is not corrupted by the input $w(k)$. These assumptions can be incorporated into System (B.1) by assuming $C_y = I$ and $D_{yw} = 0$.

The objective is to find a linear state feedback control law $u = Kx$. Following theorem gives the basis of the design.

Theorem ([2]) *There exists a controller in the form of $u(t) = Kx(t)$ such that the inequality $\|H_{wx}(s)\|_\infty^2 < \gamma$ holds if, and only if, the following LMI hold, where matrices X and L and symmetric Matrix P are the variables.*

$$M = \begin{bmatrix} P & AX + B_u L & B_w & 0 \\ (.)^T & X + X^T - P & 0 & X^T C_z^T + L^T D_{zu}^T \\ (.)^T & (.)^T & I & D_{zw}^T \\ (.)^T & (.)^T & (.)^T & \gamma I \end{bmatrix} > 0. \tag{B.2}$$

In this example we have assumed $\gamma = 1$.

The MATLAB code for this state feedback design are given in the following for the system (B.1) with $A = \begin{bmatrix} 0.01 & 0.1 \\ 0.01 & 0.2 \end{bmatrix}$, $B_w = \begin{bmatrix} 0.01 \\ 0.01 \end{bmatrix}$, $B_u = \begin{bmatrix} 0.5 \\ 1.0 \end{bmatrix}$, $C_y = [1 \quad 1]$, $C_z = [0.1 \quad 0]$, $D_{yw} = 2$, $D_{zw} = 0.1$ and $D_{zu} = 0$ are given below. After computing X, L and P, the gain matrix K is calculated from $K = LX^{-1}$.

Example_B.3.m

```
% ------------------------------------------------------------
% H-infinity state feedback control
%
clear all
clc

A=[0.01  0.1 ;0.01 0.2];
Bw= [0.01;0.1];
Bu= [0.5;1];
```

```
Cy= [1 1];
Dyw= 2;

Cz= [0.1 0];
Dzw= 0.1;
Dzu= 0;

X=sdpvar(2,2,'full');
L=sdpvar(1,2);

P=sdpvar(2,2);
gama=1;

M=[P              A*X+Bu*L          Bw           zeros(2,1);
   (A*X+Bu*L)'    X+X'-P            zeros(2,1)   X'*Cz'+L'*Dzu';
    Bw'           zeros(1,2)        eye(1)       Dzw';
    zeros(1,2)    (X'*Cz'+L'*Dzu')'  Dzw         gama*eye(1)];

Co=set(M-0.01*eye(6)>=0);

solvesdp(Co)

Xo=double(X)
Lo=double(L)
Po=double(P)
K=Lo*inv(Xo)
```

Solution

$X = [1.3217\ 0.0001; 0.0009\ 1.3358]$,
$L = [-0.0170\ \ -0.2671]$,
$P_o = [1.3131\ 0.0010; 0.0010\ 1.3407]$
and controller $K = [-0.0128\ \ -0.2000]$.

Example B.4 $\boldsymbol{H}_\infty$ Output Feedback Control

Consider System (B.1) and the following full-order linear dynamic controller

$$\begin{aligned} x_c(k+1) &= A_c x_c(k) + B_c y(k), \\ u(k) &= C_c x_c(k) + D_c y(k). \end{aligned} \tag{B.3}$$

The following theorem is the basis of this design.

Theorem ([2]) *There exists a controller in the form (B.3) such that the inequality* $\|H_{wx}(s)\|_\infty^2 < \gamma$ *holds if, and only if, the following LMI holds, where the matrices X, L, Y, F, Q, R, S, J and the symmetric matrices P and H are the variables.*

$$M = \begin{bmatrix} P & J & AX + B_uL & A + B_uRC_y & B_w + B_uRD_{yw} & 0 \\ (.)^T & H & Q & YA + FC_y & YB_w + FD_{yw} & 0 \\ (.)^T & (.)^T & X + X^T - P & I + S^T - J & 0 & X^TC_z^T + L^TD_{zu}^T \\ (.)^T & (.)^T & (.)^T & Y + Y^T - H & 0 & C_z^T + C_y^TR^TD_{zu}^T \\ (.)^T & (.)^T & (.)^T & (.)^T & I & D_{zw}^T + D_{yw}^TR^TD_{zu}^T \\ (.)^T & (.)^T & (.)^T & (.)^T & (.)^T & \gamma I \end{bmatrix} > 0 \tag{B.4}$$

Controller parameters are calculated from

$$\begin{bmatrix} A_c & B_c \\ C_c & D_c \end{bmatrix} = \begin{bmatrix} V^{-1} & -V^{-1}YB_u \\ 0 & I \end{bmatrix} \begin{bmatrix} Q - YAX & F \\ L & R \end{bmatrix} \begin{bmatrix} U^{-1} & 0 \\ -C_yXU^{-1} & I \end{bmatrix}, \tag{B.5}$$

where $UV = S - YX$.

In the example, we have assumed $\gamma = 1$.

The following code gives the H_∞-output feedback controller design for System (B.1).

Example_B.4.m

```
%_____________________________________________
% Example H-infinity output feedback

clear all
clc
gama=1;
A=[0.01  0.1 ;0.01 0.2];
Bw= [0.01;0.1];
```

```
Bu= [0.5;1];
Cy= [1 1];
Dyw= 2;
Cz= [0.1 0];
Dzw= 0.1;
Dzu= 0;
P=sdpvar(2,2);
H=sdpvar(2,2);

X=sdpvar(2,2,'full');
Y=sdpvar(2,2,'full');
J=sdpvar(2,2,'full');
S=sdpvar(2,2,'full');

F=sdpvar(2,1,'full');
Q=sdpvar(2,2,'full');
R=sdpvar(1,1,'full');
L=sdpvar(1,2,'full');

M=[P J A*X+Bu*L A+Bu*R*Cy Bw+Bu*R*Dyw zeros(2,1);
   J' H Q Y*A+F*Cy Y*Bw+F*Dyw zeros(2,1);
  (A*X+Bu*L)' Q' X+X'-P eye(2)+S'-J zeros(2,1) X'*Cz'+L'*Dzu';
  (A+Bu*R*Cy)' (Y*A+F*Cy)' (eye(2)+S'-J)' Y+Y'-H zeros(2,1) Cz'+
      Cy'*R'*Dzu';
  (Bw+Bu*R*Dyw)' (Y*Bw+F*Dyw)' zeros(1,2) zeros(1,2) eye(1) Dzw'+
      Dyw'*R'*Dzu';
   zeros(1,2) zeros(1,2) (X'*Cz'+L'*Dzu')' (Cz'+Cy'*R'*Dzu')' (
       Dzw'+Dyw'*R'*Dzu')' gama*eye(1)];

Co=set(M-0.01*eye(10)>=0);
solvesdp(Co);
X=double(X);
Y=double(Y);
P=double(P);
H=double(H);
J=double(J);
S=double(S);
Q=double(Q);
F=double(F);
R=double(R);
L=double(L);
```

```
U=eye(2);
V=S-Y*X;
Dc=R;
Bc=V\(F-Y*Bu*R);
Cc=(L-R*Cy*X)/U;
Ac=V\(Q-Y*A*X-Y*Bu*L+F*Cy*X-Y*Bu*R*Cy*X)/U;
```

Solution

$X = [1.66170\ -0.0000; 0.0067\ 1.6798]$
$Y = [1.6751\ -0.0068; -0.0322\ 1.6829]$
$P = [1.6499\ 0.0066; 0.0066\ 1.6861]$
$H = [1.6736\ 0.0001; 0.0001\ 1.6736]$
$J = [0.0075\ 0.0120; 0.0120\ 0.0210]$
$S = [-0.9755\ 0.0120; 0.0120\ -0.9790]$
$Q = 1.0e-03 \times [0.6293\ 0.0038; 0.8481\ 0.0027]$
$F = [-0.0253; -0.1027]$
$R = -0.0557$
$L = [-0.0208\ -0.3360]$
$U = [1\ 0; 0\ 1]$
$V = [-3.7590\ 0.0233; 0.0542\ -3.8058]$
$D_c = -0.0557$
$B_c = [-0.0056; 0.0025]$
$C_c = [0.0722\ -0.2423]$
$A_c = [-0.0064\ -0.0093; 0.0027\ 0.0042]$.

References

1. Lofberg J (2004) YALMIP: a tool box for modelling and optimization in matlab. In: Computer aided control systems design: 2004 IEEE international symposium, pp 284–289
2. De Oliveira MC, Geromel JC, Bernussou J (2002) Extended h_2 and h_∞ norm characterizations and controller parametrizations for discrete time systems. Int J Control 75(9):666–679

Index

A
Actuator faults, 1, 11
Adaptation law, 92, 148
Adaptive observer, 5, 88, 116

B
Bilinear matrix inequalities, 166

C
Cone complementarity linearization algorithm, 166
Coordinate transformations, 14, 36, 60, 89, 117, 147

D
Dedicated observer scheme, 12
Descriptor observer, 166
Descriptor systems, 165
Discontinuous output error injection term, 15
Disturbance attenuation, 62, 145, 150, 166

F
Fault detection, 2, 11
Fault detection and isolation, 2, 35
Fault diagnosis, 2
Fault estimation, 2, 57, 87, 115, 145, 165
Fault isolation, 2, 12
Fault-tolerant control, 2

G
Generalized observer scheme, 12
Gronwall–Bellman inequality, 18

I
Integral observer, 58, 88

L
Linear matrix inequality, 203
Lipschitz constant, 13, 88, 146
Lipschitz nonlinear systems, 6
Luenberger observer, 15, 39
Lyapunov function, 16, 40, 62, 92, 122, 150, 171

M
Matching condition, 59, 115

O
Observer-based fault diagnosis, 4

P
Popov–Belevitch–Hautus test, 38

R
Reachability condition, 19

J. Zhang et al., *Robust Observer-Based Fault Diagnosis for Nonlinear Systems Using MATLAB®*, Advances in Industrial Control,
DOI 10.1007/978-3-319-32324-4

S
Schur complement, 18, 41, 65, 204
Sensor faults, 1, 35
Sliding-mode observer, 5, 11, 35, 88, 115, 145
Sliding-mode surface, 19

T
Triangle inequality, 18

U
Unknown-input observer, 5, 145

Y
YALMIP toolbox, 213