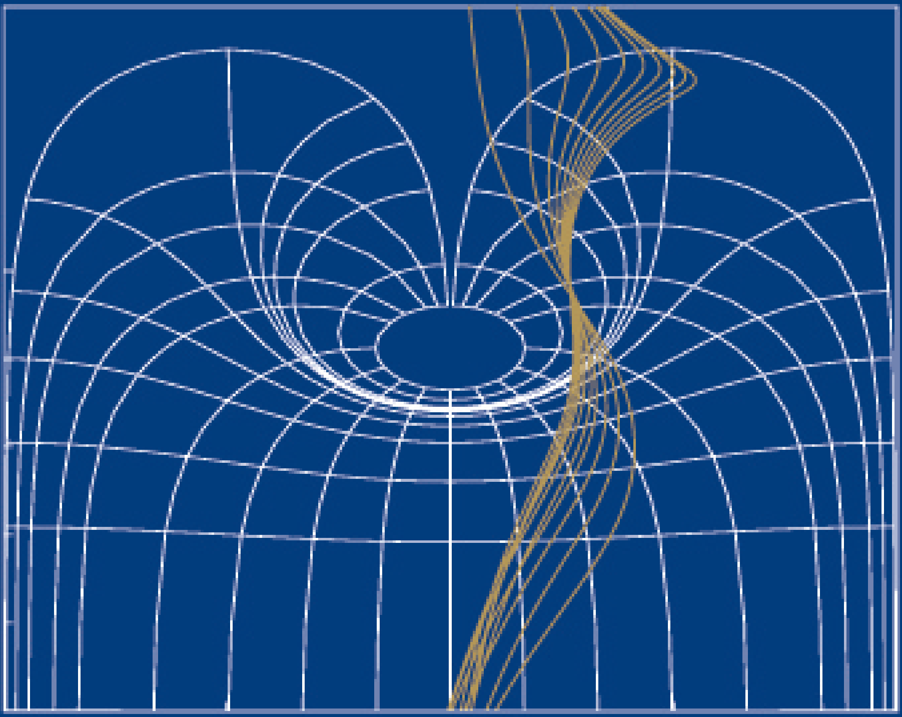
Linear Feedback Control

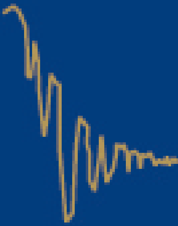
Analysis and Design with MATLAB



Dingyü Xue

YangQuan Chen

Derek P. Atherton

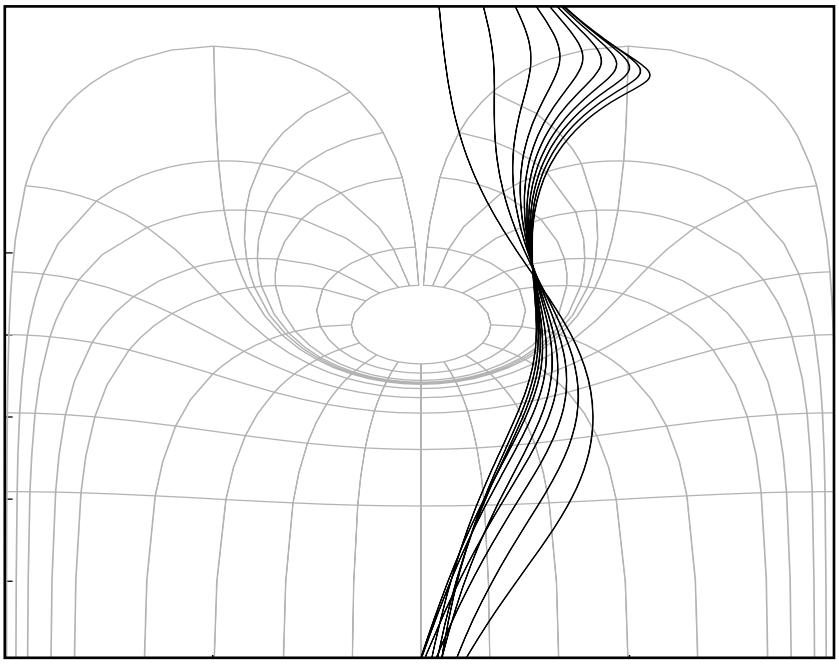


Advances in Design and Control

siam®

Linear Feedback Control

Analysis and Design with MATLAB



Advances in Design and Control

Linear Feedback Control

Analysis and Design with MATLAB

Dingyü Xue
Northeastern University
Shenyang, People's Republic of China

YangQuan Chen
Utah State University
Logan, Utah, USA

Derek P. Atherton
University of Sussex
Brighton, United Kingdom

siam®
Society for Industrial and Applied Mathematics
Philadelphia

Library of Congress Cataloging-in-Publication Data

Xue, Dingyü.
Linear feedback control : analysis and design with MATLAB / Dingyü Xue, YangQuan Chen, Derek P. Atherton.
p. cm. – (Advances in design and control)
ISBN 978-0-898716-38-2 (alk. paper)
1. Engineering mathematics—Data processing. 2. Linear control systems. 3. MATLAB. I. Chen, YangQuan. II. Atherton, Derek P. III. Title.

TA345.Z84 2007
629.8'32—dc22 2007061804

Contents

Preface

It is well known that the benefits from the wise use of control engineering are numerous and include improved product/life quality, minimized waste materials, reduced pollution, increased safety, reduced energy consumption etc. One can observe that the notions of feedback and control play important roles in most sociotechnological aspects. The phrase "control will be the physics of the 21st century"[1] implies that all engineering students should take an introductory course on systems control.

It is widely accepted that control is more "engineering" than "science," but it does require a firm theoretical underpinning for it to be successfully applied to ever more challenging projects. This attention to theory in academia has led to discussions through the years on the "theory/practice Gap" which culminated in a recent special issue of the *IEEE Control Systems Magazine* (Volume 19, Number 6, 1999).

The development of computer software for control has provided many benefits for teaching, research, and the development of control systems design in industry. MATLAB® and Simulink® are considered the dominant software platforms for control system analysis and design, with numerous off-the-shelf toolboxes dedicated to control systems and related topics. As Confucius said, "The craftsman who wishes to work well has first to sharpen his implements,"[2] and it is clear that MATLAB provides a suitable implement for control engineering. The major objective of this book is to provide information on how MATLAB can be used in control system design by covering many methods and presenting additional software routines. Many students today view control theory as difficult because of the mathematics involved in evaluating frequency responses, plotting root loci, and doing the many other calculations which can be easily accomplished in MATLAB, as shown in this book. It is therefore our opinion that the educational objective today should be to give students sufficient knowledge of these techniques to understand their relevance and teach how to use them correctly without the burden of the calculations which MATLAB can accomplish.

A distinguishing feature of the book is the organization and presentation of the material. Based on our teaching, research, and industrial experience, we have chosen to present the course materials in the following sequence: system models, time and frequency domain analysis, introduction to various model reduction techniques, model-based control design methods, PID techniques and robust control. In addition, a chapter is in-

[1]Doyle J. C. '*A new physics*?'. plenary talk presented at the 40th IEEE Conference on Decision and Control Orlando, FL, Dec. 2001.

[2]`http://www.confucius.org/lunyu/ed1509.htm.`

cluded on fractional-order control as an alternative for practical robustness trade-offs. MATLAB scripts and plots are extensively used in this textbook to illustrate basic concepts and examples. A dedicated toolbox called CtrlLAB developed by the authors can be used as an effective teaching and learning aid. CtrlLAB was developed to support our objective of enabling control studies to be done in MATLAB by students with no knowledge of MATLAB, thus avoiding the need to replace less mathematics with the requirement of learning a programming language (although this is not difficult). CtrlLAB is the most downloaded package in the Control Systems category in the File Exchange of MATLAB Central.[3]

We hope that readers will enjoy playing with and changing the scripts as they gain better understanding and accomplish deeper exploration with reduced effort. Additionally, each chapter comes with a set of problems to strengthen the readers' understanding of the chapter contents.

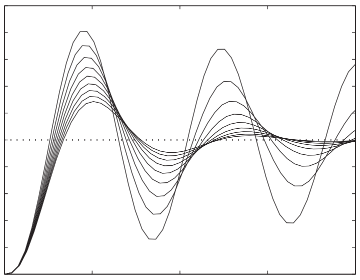
This book can be used as a reference text in the introductory control course for undergraduates in all engineering schools. The coverage of topics is broad, yet balanced, and should provide a solid foundation for the subsequent control engineering practice in both industry and research institutes. For graduates and researchers not majoring in control, this textbook is useful for knowledge enhancement. The authors also believe that this book will be a good desktop reference for control engineers.

The writing of this book started in the mid 1990s. In its evolving into the current form, many researchers, professors, and students have provided useful feedback, comments, and input. In particular, we thank the following professors: Xinhe Xu, Xingquan Ren, Yuanwei Jing, Taicheng Yang, Shuzhi Sam Ge, Igor Podlubny, Ivo Petras, István Kollár, Alain Oustaloup, Jocelyn Sabatier, Blas M. Vinagre, J. A. Tenreiro Machado, and Kevin L. Moore. Moreover, we are grateful to Elizabeth Greenspan, Acquisitions Editor of the Society for Industrial and Applied Mathematics (SIAM), for her professional help. The "Book Program" from The MathWorks Inc. is acknowledged for the latest MATLAB software.

Last, but not least, Dingyü Xue would like to thank his wife Jun Yang and his daughter Yang Xue; YangQuan Chen would like to thank his wife Huifang Dou and his sons Duyun, David, and Daniel, for their patience, understanding and complete support throughout this work. Derek Atherton wishes to thank his wife Constance for allowing him hours of overtime with many hardworking graduate students which included, in particular, many discussions with Dingyü when he was at Sussex and the email exchanges or with Dingyü and YangQuan, which led to this book.

Dingyü Xue, Northeastern University, Shenyang, China.
YangQuan Chen, Utah State University, Logan, UT, USA.
Derek P. Atherton, The University of Sussex, Brighton, UK.

[3]`http://www.mathworks.com/matlabcentral/index.shtml`



Chapter 8

Fractional-Order Controller: An Introduction

Using the notion of fractional-order may be a more realistic step because real processes are generally “fractional” [86]. However, for many real processes, fractionality is very low. A typical example of a noninteger, (fractional-) order system is the voltage–current relationship of a semi-infinite lossy resistor and capacitor (RC) line or the diffusion of heat in a semi-infinite solid, where the heat flow $q(t)$ is naturally equal to the semiderivative of temperature $T(t)$ [87], as described by the following simple fractional-order differential equation (FODE):

$$\frac{\mathrm{d}^{0.5}T(t)}{\mathrm{d}t^{0.5}} = q(t).$$

Clearly, using an integer-order ordinary differential equation (ODE) description for the above system may differ significantly from the actual situation. However, the fact that the integer-order dynamic models are more welcome is probably due to the absence of solution methods for FODEs. Details of past and present progress in the analysis of dynamic systems modeled by FODEs can be found in [88–95]. For example, PID (proportional integral derivative) controllers, which have been dominating industrial controllers, have been modified using the notion of a fractional-order integrator and differentiator. It has been shown that two extra degrees of freedom from the use of a fractional-order integrator and differentiator make it possible to further improve the performance of traditional PID controllers. In addition, the plant to be controlled can also be modeled as a dynamic system described by an FODE. For fractional-order systems, the fractional controller CRONE was developed in [96], while [89, 97, 98] presented the PD$^{\delta}$ controller and [99] proposed the PI$^{\lambda}$D$^{\delta}$ controller.

In theory, control systems can include both the fractional-order dynamic system or plant to be controlled and the fractional-order controller. However, in control engineering, it is a common practice to consider only the fractional-order controller. This is due to the fact that the plant model may have already been obtained as an integer-order model in a classical sense. In most cases, our objective is to apply fractional-order control (FOC) to enhance system control performance. Therefore, in this chapter we will concentrate on the scenario in which the controller is fractional-order.

This chapter serves as an introduction to the essentials of FOC for control engineering practice, with an emphasis on how to analyze and realize fractional-order systems using MATLAB. For a broader introductory coverage of fractional-order calculus and its applications in engineering, we refer the interested reader to the textbook [100].

This chapter is organized as follows. In Sec. 8.1, definitions and properties of fractional-order calculus are briefly introduced, followed by frequency and time domain analysis of fractional-order linear systems in Sec. 8.2. Then, in Sec. 8.3 filter approximations to fractional-order differentiators are introduced using Oustaloup's recursive scheme and its refined version. With this filter approximation, using Simulink, a simulation method for a general nonlinear fractional-order dynamic system is proposed with an illustrative example. Since the fractional-order controller after finite dimensional approximation is usually of a very high order, controller order reduction is discussed and demonstrated in Sec. 8.4. Finally, we present some controller design case studies for fractional-order systems in Sec. 8.5.

Note that this chapter, like previous chapters, is designed so that the text and illustrative MATLAB scripts flow in a natural and smooth manner. We hope that this design enables readers to quickly get started on problem solving. It is worth mentioning that the design of a MATLAB class for a fractional-order transfer function is demonstrated thoroughly in the chapter.

8.1 Fractional-Order Calculus and Its Computations

In a letter to Hôpital in 1695, Leibniz raised the following question: Can the meaning of derivatives with integer order $\mathrm{d}^n y(x)/\mathrm{d}x^n$ be generalized to derivatives with noninteger orders, so that in general $n \in \mathscr{C}$? (Here $\mathscr{C}$ is the set for all complex numbers.) Hôpital was a bit curious about this question and replied with another question to Leibniz: What if $n = 1/2$? Leibniz, in a letter dated September 30, 1695, replied: It will lead to a paradox, from which one day useful consequences will be drawn.

The question raised by Leibniz for a fractional-order derivative has been a topic of ongoing study in the last 300 years. Several mathematicians contributed to this subject over the years. People like Liouville, Riemann, and Weyl made major contributions to the theory of fractional-order calculus. So, the term "fractional-order calculus" is by no means new. It is a generalization of ordinary differentiation by noninteger derivatives. The subject is as old as the calculus of differentiation and goes back to the 17th century when Leibniz and Newton invented calculus. The theory of fractional-order derivatives was developed mainly in the 19th century. For more information, see [91, 93, 101, 102].

In the development of fractional-order calculus, there appeared different definitions of fractional-order differentiations and integrations. Some of the definitions extend directly from integer-order calculus. The well-established definitions include the Cauchy integral formula, the Grünwald–Letnikov definition, the Riemann–Liouville definition, and the Caputo definition. The definitions will be summarized first, and then properties will be given.

8.1.1 Definitions of Fractional-Order Calculus

Definition 8.1 (*Cauchy's fractional-order integration formula*). This definition is a general extension of the integer-order Cauchy formula

$$\mathscr{D}^{\gamma} f(t) = \frac{\Gamma(\gamma+1)}{2\pi \mathrm{j}} \int_C \frac{f(\tau)}{(\tau-t)^{\gamma+1}} \mathrm{d}\tau, \tag{8.1}$$

where C is the smooth curve encircling the single-valued function $f(t)$.

Definition 8.2 (*Grünwald–Letnikov definition*). The definition is defined as

$${}_a\mathscr{D}_t^{\alpha} f(t) = \lim_{h\to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{[(t-a)/h]} (-1)^j \binom{\alpha}{j} f(t-jh), \tag{8.2}$$

where $w_j^{(\alpha)} = (-1)^j \binom{\alpha}{j}$ represents the coefficients of the polynomial $(1-z)^{\alpha}$. The coefficients can also be obtained recursively from

$$w_0^{(\alpha)} = 1, \;\; w_j^{(\alpha)} = \left(1 - \frac{\alpha+1}{j}\right) w_{j-1}^{(\alpha)}, \;\; j = 1, 2, \ldots. \tag{8.3}$$

Based on the Definition 8.2, the fractional-order differentiation can easily be calculated from

$${}_a\mathscr{D}_t^{\alpha} f(t) = \lim_{h\to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{[(t-a)/h]} (-1)^j \binom{\alpha}{j} f(t-jh) \approx \frac{1}{h^{\alpha}} \sum_{j=0}^{[(t-a)/h]} w_j^{(\alpha)} f(t-jh). \tag{8.4}$$

Assuming that the step size h is small enough, we see that (8.4) can be used to evaluate the differentiations of the given function. It can be shown [93] that the accuracy of the method is $\mathrm{o}(h)$. Thus, based on the Grünwald–Letnikov definition, the following MATLAB function can be written to evaluate the fractional-order differentiation [103]:

```
function dy=glfdiff(y,t,gam)
h=t(2)-t(1); dy(1)=0; y=y(:); t=t(:);
w=1; for j=2:length(t), w(j)=w(j-1)*(1-(gam+1)/(j-1)); end
for i=2:length(t), dy(i)=w(1:i)*[y(i:-1:1)]/h^gam; end
```

The syntax of the function is `dy=glfdiff(y,t,γ)` ($\boldsymbol{d}_y$=`glfdiff(`$\boldsymbol{y}$`,`$\boldsymbol{t}$`,`γ`)`), where $\boldsymbol{y}, \boldsymbol{t}$ are, respectively, the vectors composed of the samples and the time instances. The time vector $\boldsymbol{t}$ is assumed to be evenly distributed. γ is the order of fractional-order differentiation. The returned vector $\boldsymbol{d}_y$ is the vector of the fractional-order derivatives.

Definition 8.3 (*Riemann–Liouville fractional-order differentiation*). The fractional-order integration is defined as

$${}_a\mathscr{D}_t^{-\alpha} f(t) = \frac{1}{\Gamma(\alpha)} \int_a^t (t-\tau)^{\alpha-1} f(\tau) \mathrm{d}\tau, \tag{8.5}$$

where $0 < \alpha < 1$, and a is the initial time instance, often assumed to be zero, i.e., $a = 0$. The differentiation is then denoted as $\mathscr{D}_t^{-\alpha} f(t)$.

The Riemann–Liouville definition is the most widely used definition in fractional-order calculus. The subscripts on both sides of $\mathscr{D}$ represent, respectively, the lower and upper bounds in the integration [104].

Such a definition can also be extended to fractional-order differentiations when the order satisfies $n-1 < \beta \leq n$. The fractional-order differentiation is then defined as

$$ {}_a\mathscr{D}_t^{\beta} f(t) = \frac{\mathrm{d}^n}{\mathrm{d}t^n}\left[{}_a\mathscr{D}_t^{-(n-\beta)} f(t)\right] = \frac{1}{\Gamma(n-\beta)}\frac{\mathrm{d}^n}{\mathrm{d}t^n}\left[\int_a^t \frac{f(\tau)}{(t-\tau)^{\beta-n+1}}\mathrm{d}\tau\right]. \tag{8.6}$$

Definition 8.4 (*Caputo's definition of fractional-order differentiation*). Caputo's definition is given by

$$ {}_0\mathscr{D}_t^{\alpha} y(t) = \frac{1}{\Gamma(1-\gamma)}\int_0^t \frac{y^{(m+1)}(\tau)}{(t-\tau)^{\gamma}}\mathrm{d}\tau, \tag{8.7}$$

where $\alpha = m + \gamma$, m is an integer, and $0 < \gamma \leq 1$. Similarly, Caputo's fractional-order integration is defined as

$$ {}_0\mathscr{D}_t^{\gamma} = \frac{1}{\Gamma(-\gamma)}\int_0^t \frac{y(\tau)}{(t-\tau)^{1+\gamma}}\mathrm{d}\tau, \quad \gamma < 0. \tag{8.8}$$

It can be shown [93] that for a class of real functions, the fractional-order differentiations from the Grünwald–Letnikov and Riemann–Liouville definitions are identical.

8.1.2 Properties of Fractional-Order Differentiations

The fractional-order differentiation has the following properties [105]:

1. The fractional-order differentiation ${}_0\mathscr{D}_t^{\alpha} f(t)$, with respect to t of an analytic function $f(t)$, is also analytical.
2. The fractional-order differentiation is exactly the same with integer-order one, when $\alpha = n$ is an integer. Also ${}_0\mathscr{D}_t^{0} f(t) = f(t)$.
3. The fractional-order differentiation is linear; i.e., for any constants a, b, one has
$$ {}_0\mathscr{D}_t^{\alpha}\left[af(t) + bg(t)\right] = a\,{}_0\mathscr{D}_t^{\alpha} f(t) + b\,{}_0\mathscr{D}_t^{\alpha} g(t). \tag{8.9}$$
4. Fractional-order differentiation operators satisfy the commutative-law, and also satisfy
$$ {}_0\mathscr{D}_t^{\alpha}\left[{}_0\mathscr{D}_t^{\beta} f(t)\right] = {}_0\mathscr{D}_t^{\beta}\left[{}_0\mathscr{D}_t^{\alpha} f(t)\right] = {}_0\mathscr{D}_t^{\alpha+\beta} f(t) \tag{8.10}$$
5. The Laplace transform of fractional-order differentiation is defined as
$$ \mathscr{L}\left[{}_0\mathscr{D}_t^{\alpha} f(t)\right] = s^{\alpha}\mathscr{L}[f(t)] - \sum_{k=1}^{n-1} s^k \left[{}_0\mathscr{D}_t^{\alpha-k-1} f(t)\right]_{t=0}. \tag{8.11}$$
In particular, if the derivatives of the function $f(t)$ are all equal to 0 at $t = 0$, one has $\mathscr{L}[{}_0\mathscr{D}_t^{\alpha} f(t)] = s^{\alpha}\mathscr{L}[f(t)]$.

8.2 Frequency and Time Domain Analysis of Fractional-Order Linear Systems

The fractional-order system is the direct extension of classical integer-order systems. The fractional-order system is established upon the fractional-order differential equations, and the fractional-order transfer function of a single variable system can be defined as

$$G(s) = \frac{b_1 s^{\gamma_1} + b_2 s^{\gamma_2} + \cdots + b_m s^{\gamma_m}}{a_1 s^{\eta_1} + a_2 s^{\eta_2} + \cdots + a_{n-1} s^{\eta_{n-1}} + a_n s^{\eta_n}}, \tag{8.12}$$

where b_i, a_i are real numbers and the orders γ_i, η_i of the numerator and the denominator can also be real numbers. The analysis of the fractional-order Laplace transformations and their inverse is very complicated. The closed-form solutions to the problems are not possible in general.

8.2.1 Fractional-Order Transfer Function Modeling

For the fractional-order transfer function model in (8.12), it can be seen that if the coefficients and the orders of the numerator and denominator are given, the model can be established. Thus, an "`fotf`" class can be constructed by creating the `@fotf` directory and writing in the directory an `fotf()` function as follows:

```
function G=fotf(a,na,b,nb)
if nargin==0,
   G.a=[]; G.na=[]; G.b=[]; G.nb=[]; G=class(G,'fotf');
elseif isa(a,'fotf'), G=a;
elseif nargin==1 & isa(a,'double'), G=fotf(1,0,a,0);
else,
   ii=find(abs(a)<eps); a(ii)=[]; na(ii)=[];
   ii=find(abs(b)<eps); b(ii)=[]; nb(ii)=[];
   G.a=a; G.na=na; G.b=b; G.nb=nb; G=class(G,'fotf');
end
```

The syntax of the function is G=`fotf(`$\boldsymbol{a}$`,`$\boldsymbol{\eta}$`,`$\boldsymbol{b}$`,`$\boldsymbol{\gamma}$`)`, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are the coefficients of the denominator and the numerator, respectively, while $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ are the order sequences in the denominator and the numerator, respectively.

A display function should also be created for the `fotf` class. The file should also be saved in the `@fotf` directory such that

```
function display(G)
sN=polydisp(G.b,G.nb); sD=polydisp(G.a,G.na); s=' ';
nm=max([length(sN),length(sD)]); nn=length(sN); nd=length(sD);
disp([char(s*ones(1,floor((nm-nn)/2))) sN]), disp(char('-'*ones(1,nm)));
disp([char(s*ones(1,floor((nm-nd)/2))) sD])
function strP=polydisp(p,np)
P=''; [np,ii]=sort(np,'descend'); p=p(ii);
for i=1:length(p), P=[P,'+',num2str(p(i)),'s^{',num2str(np(i)),'}']; end
P=P(2:end); P=strrep(P,'s^{0}',''); P=strrep(P,'+-','-');
P=strrep(P,'^{1}',''); P=strrep(P,'+1s','+s'); strP=strrep(P,'-1s','-s');
if length(strP)>=2, if strP(1:2)=='1s', strP=strP(2:end); end,end,
```

Example 8.1. Suppose that the fractional-order transfer function is given by

$$G(s) = \frac{-2s^{0.63} - 4}{2s^{3.501} + 3.8s^{2.42} + 2.6s^{1.798} + 2.5s^{1.31} + 1.5}.$$

With the following statement, the fractional-order transfer function can be entered into the MATLAB environment:

```
>> b=[-2,-4]; nb=[0.63,0]; a=[2 3.8 2.6 2.5 1.5];
   na=[3.501,2.42,1.798,1.31,0]; G=fotf(a,na,b,nb)
```

The display of the fractional-order transfer function is

```
                        -2s^{0.63}-4
--------------------------------------------------
2s^{3.501}+3.8s^{2.42}+2.6s^{1.798}+2.5s^{1.31}+1.5
```

A function `fotf()` can be written in the `@tf` directory to convert an integer-order transfer function to an `fotf` object:

```
function G1=fotf(G)
n=G.num{1}; d=G.den{1}; i1=find(abs(n)<eps); i2=find(abs(d)<eps);
if length(i1)>0 & i1(1)==1, n=n(i1(1)+1:end); end
if length(i2)>0 & i2(1)==1, d=d(i2(1)+1:end); end
G1=fotf(d,length(d)-1:-1:0,n,length(n)-1:-1:0);
```

8.2.2 Interconnections of Fractional-Order Blocks

Based on the newly defined `fotf` class, the `plus()`, `mtimes()` and `feedback()` functions can be written as follows:

- Plus function `plus()` for block parallel connections:

```
function G=plus(G1,G2)
a=kron(G1.a,G2.a); b=[kron(G1.a,G2.b), kron(G1.b,G2.a)]; na=[]; nb=[];
for i=1:length(G1.a), na=[na G1.na(i)+G2.na]; nb=[nb, G1.na(i)+G2.nb]; end
for i=1:length(G1.b), nb=[nb G1.nb(i)+G2.na]; end
G=unique(fotf(a,na,b,nb));
```

- Multiplication function `mtimes()` for block series connections:

```
function G=mtimes(G1,G2)
G2=fotf(G2); a=kron(G1.a,G2.a);
b=kron(G1.b,G2.b); na=[]; nb=[];
for i=1:length(G1.na), na=[na,G1.na(i)+G2.na]; end
for i=1:length(G1.nb), nb=[nb,G1.nb(i)+G2.nb]; end
G=unique(fotf(a,na,b,nb));
```

- Feedback function `feedback()` for block negative feedback connections:

```
function G=feedback(F,H)
H=fotf(H);
```

```
b=kron(F.b,H.a); a=[kron(F.b,H.b), kron(F.a,H.a)]; na=[]; nb=[];
for i=1:length(F.b), nb=[nb F.nb(i)+H.nb]; na=[na,F.nb(i)+H.nb]; end
for i=1:length(F.a), na=[na F.na(i)+H.na]; end
G=unique(fotf(a,na,b,nb));
```

- Simplification function `unique()`:

```
function G=unique(G)
[a,n]=polyuniq(G.a,G.na); G.a=a; G.na=n;
[a,n]=polyuniq(G.b,G.nb); G.b=a; G.nb=n;
function [a,an]=polyuniq(a,an)
[an,ii]=sort(an,'descend'); a=a(ii); ax=diff(an); key=1;
for i=1:length(ax)
   if ax(i)==0, a(key)=a(key)+a(key+1); a(key+1)=[]; an(key+1)=[];
   else, key=key+1; end
end
```

Other functions should also be designed, such as `minus()`, `uminus()`, `inv()`, and the files should be placed in the `@fotf` directory to overload the existing ones. The listings of these functions are not given in this text but available from the book's companion Website.

Example 8.2. Suppose in the unity negative feedback system, the system models are given by

$$G(s) = \frac{0.8s^{1.2}+2}{1.1s^{1.8}+0.8s^{1.3}+1.9s^{0.5}+0.4},\ G_c(s) = \frac{1.2s^{0.72}+1.5s^{0.33}}{3s^{0.8}}.$$

The plant and controller can be easily entered and the closed-loop system can be directly obtained with the commands

```
>> G=fotf([1.1,0.8 1.9 0.4],[1.8 1.3 0.5 0],[0.8 2],[1.2 0]);
   Gc=fotf(3,[0.8],[1.2 1.5],[0.72 0.33]); H=fotf(1,0,1,0);
   GG=feedback(G*Gc,H)
```

and the result is given by

$$G(s) = \frac{0.96s^{1.92}+1.2s^{1.53}+2.4s^{0.72}+3s^{0.33}}{3.3s^{2.6}+2.4s^{2.1}+0.96s^{1.92}+1.2s^{1.53}+5.7s^{1.3}+1.2s^{0.8}+2.4s^{0.72}+3s^{0.33}}.$$

It can be seen from the above illustrations that, although the plant and controllers are relatively simple, an extremely complicated closed-loop model may be obtained. This makes the analysis and design of the fractional-order system a difficult task.

8.2.3 Frequency Domain Analysis of Linear Fractional-Order Systems

It can be seen that, when $\mathrm{j}\omega$ is used to substitute for the variable s in the fractional-order transfer function model, the frequency domain response $G(\mathrm{j}\omega)$ can be easily evaluated. Thus, the fractional-order Bode diagrams, Nyquist plots, and Nichols charts can be easily evaluated with the function `bode()`, which is written as an overload function for the `fotf` object

```
function H=bode(G,w)
a=G.a; eta=G.na; b=G.b; g=G.nb; if nargin==1, w=logspace(-4,4); end
for i=1:length(w)
    P=b*((sqrt(-1)*w(i)).^g.'); Q=a*((sqrt(-1)*w(i)).^eta.'); H1(i)=P/Q;
end
H1=frd(H1,w); if nargout==0, bode(H1); else, H=H1; end
```

The syntax of the function is `H=bode(G,ω)`, where G is the fractional-order transfer function object and the optional argument $\boldsymbol{\omega}$ is the frequency vector.

If one wants to draw the Bode diagram, there is no need to return any variable. If frequency domain response data are needed, the response results can be found in the returned variable H. The variable H can be used in drawing the Nyquist plot and the Nichols chart by using `nyquist(H)` and `nichols(H)`, respectively.

8.2.4 Time Domain Analysis of Fractional-Order Systems

The evaluation of the time domain response of a fractional-order system is more complicated. Let us consider a special form of a fractional-order differential equation [93]

$$a_1 \mathscr{D}_t^{\eta_1} y(t) + a_2 \mathscr{D}_t^{\eta_2} y(t) + \cdots + a_{n-1} \mathscr{D}_t^{\eta_{n-1}} y(t) + a_n \mathscr{D}_t^{\eta_n} y(t) = u(t), \tag{8.13}$$

where $u(t)$ can be represented by a certain function and its fractional-order derivatives. Assume also that the output function $y(t)$ has zero initial conditions. The Laplace transform can be used to find the transfer function

$$G(s) = \frac{1}{a_1 s^{\eta_1} + a_2 s^{\eta_2} + \cdots + a_{n-1} s^{\eta_{n-1}} + a_n s^{\eta_n}}. \tag{8.14}$$

Consider the Grünwald–Letnikov definition in (8.4). The discrete form of it can be rewritten as

$${}_a\mathscr{D}_t^{\eta_i} y(t) \simeq \frac{1}{h^{\eta_i}} \sum_{j=0}^{[(t-a)/h]} w_j^{(\eta_i)} y_{t-jh} = \frac{1}{h^{\eta_i}} \left[y_t + \sum_{j=1}^{[(t-a)/h]} w_j^{(\eta_i)} y_{t-jh} \right], \tag{8.15}$$

where $w_0^{(\beta_i)}$ can be evaluated recursively from the formula (8.3). By substituting it into (8.13), the numerical solution to the fractional-order differential equation can be written as

$$y_t = \frac{1}{\sum_{i=1}^{n} \frac{a_i}{h^{\eta_i}}} \left[u_t - \sum_{i=1}^{n} \frac{a_i}{h^{\eta_i}} \sum_{j=1}^{[(t-a)/h]} w_j^{(\eta_i)} y_{t-jh} \right]. \tag{8.16}$$

For the general form of the fractional-order transfer function in (8.12), the right-hand side can equivalently be evaluated first by using the numerical method discussed earlier. The final solution can be obtained from (8.16). A MATLAB function can be written for the `fotf` object to evaluate the time domain response as follows:

```
function y=lsim(G,u,t)
a=G.a; eta=G.na; b=G.b; gamma=G.nb; nA=length(a);
h=t(2)-t(1); D=sum(a./[h.^eta]); W=[]; nT=length(t);
```

```
vec=[eta gamma]; D1=b(:)./h.^gamma(:);
y1=zeros(nT,1); W=ones(nT,length(vec));
for j=2:nT, W(j,:)=W(j-1,:).*(1-(vec+1)/(j-1)); end
for i=2:nT
   A=[y1(i-1:-1:1)]'*W(2:i,1:nA); y1(i)=(u(i)-sum(A.*a./[h.^eta]))/D;
end
for i=2:nT, y(i)=(W(1:i,nA+1:end)*D1)'*[y1(i:-1:1)]; end
```

The syntax of the function is `y=lsim(G,u,t)`, where the time vector and the input vector are defined in the variables $\boldsymbol{t}$ and $\boldsymbol{u}$, respectively. The returned vector $\boldsymbol{y}$ is the solution to the equations. If there are more points in the equation, the computation may be very slow.

An overloaded `step()` function can also be written, based on the `lsim()` function given above, as

```
function y=step(G,t)
u=ones(size(t)); y=lsim(G,u,t);
if nargout==0, plot(t,y); end
```

with `y=step(G,t)`, where G is an `fotf` object, and $\boldsymbol{t}$ should be given as an evenly distributed time vector. The step response of the system is returned in vector $\boldsymbol{y}$.

It is possible to solve the above fractional-order differential equation analytically by using the Mittag–Leffler function in two parameters, which is a generalization of the exponential function e^z. The Mittag–Leffler function in two parameters is defined as

$$\mathscr{E}_{\alpha,\beta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k+\beta)}, \quad (\alpha, \beta > 0). \tag{8.17}$$

Clearly, e^z is a particular case of the Mittag–Leffler function [92]:

$$\mathscr{E}_{1,1}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(k+1)} = \sum_{k=0}^{\infty} \frac{z^k}{k!} = \mathrm{e}^z.$$

Furthermore, one can get more particular cases for the Mittag–Leffler function in two parameters, for example,

$$\mathscr{E}_{2,1}(z) = \cosh(\sqrt{z}), \quad \mathscr{E}_{1,2}(z) = \frac{\mathrm{e}^z - 1}{z}, \quad \mathscr{E}_{2,2}(z) = \frac{\sinh(\sqrt{z})}{\sqrt{z}}, \tag{8.18}$$

$$\mathscr{E}_{1/2,1}(\sqrt{z}) = \frac{2}{\sqrt{\pi}}\mathrm{e}^{-z}\mathrm{erfc}(-\sqrt{z}). \tag{8.19}$$

The analytical solution of the n-term FODE is given in general form [92] by

$$y(t) = \frac{1}{a_n}\sum_{m=0}^{\infty}\frac{(-1)^m}{m!} \sum_{\substack{k_0+k_1+\cdots+k_{n-2}=m \\ k_0\geq 0,\ldots,k_{n-2}\geq 0}} (m; k_0, k_1, \ldots, k_{n-2})$$

$$\prod_{i=0}^{n-2}\left(\frac{a_i}{a_n}\right)^{k_i} t^{(\beta_n-\beta_{n-1})m+\beta_n+\sum\limits_{j=0}^{n-2}(\beta_{n-1}-\beta_j)k_j-1} \tag{8.20}$$

$$\mathscr{E}^{(m)}_{\beta_n-\beta_{n-1},\beta_n+\sum_{j=0}^{n-2}(\beta_{n-1}-\beta_j)k_j}\left(-\frac{a_{n-1}}{a_n}t^{\beta_n-\beta_{n-1}}\right),$$

where $\mathscr{E}_{\lambda,\mu}(z)$ is the Mittag–Leffler function in two parameters as defined in (8.17) and

$$\mathscr{E}^{(n)}_{\lambda,\mu}(y) \equiv \frac{\mathrm{d}^n}{\mathrm{d}\,y^n}\mathscr{E}_{\lambda,\mu}(y) = \sum_{j=0}^{\infty}\frac{(j+n)!\,y^j}{j!\,\Gamma(\lambda j+\lambda n+\mu)} \text{ for } n = 0, 1, 2, \dots . \tag{8.21}$$

8.3 Filter Approximation to Fractional-Order Differentiations

It can be seen that the Grünwald–Letnikov definition gives a very good fitting to the fractional-order derivatives for given functions. However, in control system analysis and design, the definition is not useful, since the samples of the function should be known. Online real-time fractional-order differentiation may be required in control systems. Using filters is one of the best ways to solve the problems.

8.3.1 Oustaloup's Recursive Filter

Some continuous filters have been summarized in [105]. Among the filters, the well-established Oustaloup recursive filter has a very good fitting to the fractional-order differentiators [106]. Assume that the expected fitting range is (ω_b, ω_h). The filter can be written as

$$G_f(s) = K\prod_{k=-N}^{N}\frac{s+\omega_k'}{s+\omega_k}, \tag{8.22}$$

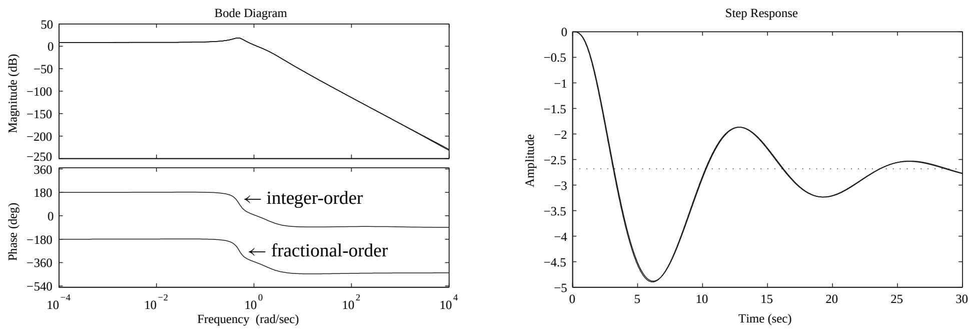
where the poles, zeros, and gain of the filter can be evaluated from (8.23) such that

$$\omega_k' = \omega_b\left(\frac{\omega_h}{\omega_b}\right)^{\frac{k+N+\frac{1}{2}(1-\gamma)}{2N+1}},\quad \omega_k = \omega_b\left(\frac{\omega_h}{\omega_b}\right)^{\frac{k+N+\frac{1}{2}(1+\gamma)}{2N+1}},\quad K = \omega_h^{\gamma}. \tag{8.23}$$

With the above algorithm, the following MATLAB function `oustafod()` can be written to design the continuous filter. Thus, the $y(t)$ signal can be filtered through the filter and the output of the filter can be regarded as an approximation to the $\mathscr{D}_t^{\gamma}\,y(t)$ signal.

```
function G=oustafod(r,N,wb,wh)
mu=wh/wb; k=-N:N; w_kp=(mu).^((k+N+0.5-0.5*r)/(2*N+1))*wb;
w_k=(mu).^((k+N+0.5+0.5*r)/(2*N+1))*wb;
K=wh^r; G=tf(zpk(-w_kp',-w_k',K));
```

The function can be called with G_f`=oustafod(`γ`,`N`,`ω_b`,`ω_h`)`, where γ is the order of the differentiation, $2N+1$ is the order of the filter, and the frequency fitting range is given by (ω_b, ω_h). The filter G_f can be designed such that it may fit very well within the frequency range of the fractional order differentiator.



(a) Bode diagrams (b) step response

Figure 8.1. *Time and frequency domain comparisons.*

Example 8.3. Consider a fractional-order model

$$G(s) = \frac{-2s^{0.63} - 4}{2s^{3.501} + 3.8s^{2.42} + 2.6s^{1.798} + 2.5s^{1.31} + 1.5}.$$

Since the original orders are all fractional, it may not be easy to design controllers for them. Thus, a model reduction technique can be considered to reduce the order such that a low integer-order approximation can be achieved. Suppose that one wants to approximate the differentiators within the frequency range of $(10^{-3}, 10^4)$; the high-order term can also be approximated as $s^{3.501} = s^3 s^{0.501}$, and the integer-order approximation can be obtained as

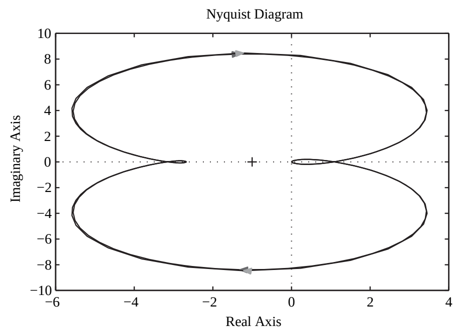
```
>> N=4; w1=1e-3; w2=1e4; g1=oustafod(0.501,N,w1,w2);
   s=tf('s');
   g2=oustafod(0.42,N,w1,w2); g3=oustafod(0.798,N,w1,w2);
   g4=oustafod(0.31,N,w1,w2); g5=oustafod(0.63,N,w1,w2);
   G1=(-2*g5-4)/(2*s^3*g1+3.8*s^2*g2+2.6*s*g3+2.5*s*g4+1.5);
```

It is found that the order of the approximation reaches 48. The exact Bode diagram and its 48th-order approximation are shown in Figure 8.1(a). The step responses of the system is obtained as shown in Figure 8.1(b). With the following MATLAB statements, it can be seen that the time response of the filter can accurately approximate the fractional-order derivatives of the system.

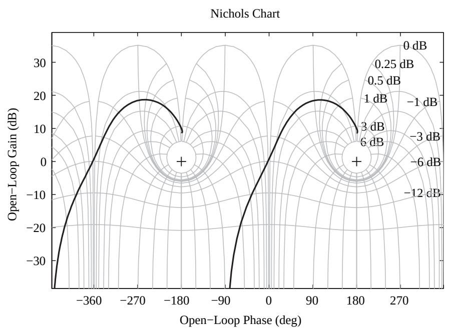
```
>> b=[-2 -4]; nb=[0.63 0]; a=[2 3.8 2.6 2.5 1.5];
   na=[3.501 2.42 1.798 1.31 0]; G=fotf(a,na,b,nb);
   w=logspace(-4,4,500); H=bode(G,w); bode(G1,H,{1e-4,1e4});
   figure; t=0:0.004:30; y=step(G,t); step(G1,30); line(t,y)
```

The open-loop Nyquist plots and Nichols charts can also be obtained as shown in Figure 8.2. It can be seen that the Nyquist plot accurately fits the theoretical one, while the Nichols chart is shifted by 360°, which means the two are identical:

```
>> H=bode(G,w); nyquist(G,H,{1e-4,1e4});
   figure; nichols(G,H,{1e-4,1e4}); grid
```



(a) Nyquist plots



(b) Nichols plots

Figure 8.2. *Comparisons of other frequency domain plots.*

8.3.2 A Refined Oustaloup Filter

Here we introduce a new approximate realization method for the fractional-order derivative in the frequency range of interest $[\omega_b, \omega_h]$. Our proposed method here gives a better approximation than Oustaloup's method with respect to both low frequency and high frequency.

Assume that the frequency range to be fit is defined as (ω_b, ω_h). Within the prespecified frequency range, the fractional-order operator s^α can be approximated by the fractional-order transfer function as

$$K(s) = \left(\frac{1 + \frac{bs}{d\omega_b}}{1 + \frac{ds}{b\omega_h}} \right)^\alpha , \tag{8.24}$$

where $0 < \alpha < 1,\ s = \mathrm{j}\omega,\ b > 0,\ d > 0$, and

$$K(s) = \left(\frac{bs}{d\omega_b} \right)^\alpha \left(1 + \frac{-ds^2 + d}{ds^2 + b\omega_h s} \right)^\alpha . \tag{8.25}$$

In the frequency range $\omega_b < \omega < \omega_h$, by using a Taylor series expansion, we obtain

$$K(s) = \left(\frac{bs}{d\omega_b} \right)^\alpha \left(1 + \alpha p(s) + \frac{\alpha(\alpha - 1)}{2} p^2(s) + \cdots \right) \tag{8.26}$$

with

$$p(s) = \frac{-ds^2 + d}{ds^2 + b\omega_h s}.$$

It is then found that

$$s^\alpha = \frac{(d\omega_b)^\alpha b^{-\alpha}}{\left[1 + \alpha p(s) + \frac{\alpha(\alpha - 1)}{2} p^2(s) + \cdots \right]} \left(\frac{1 + \frac{bs}{d\omega_b}}{1 + \frac{ds}{b\omega_h}} \right)^\alpha . \tag{8.27}$$

Truncating the Taylor series to 1 leads to

$$s^\alpha \approx \frac{(d\omega_b)^\alpha}{b^\alpha\big(1+\alpha p(s)\big)} \left(\frac{1+\frac{bs}{d\omega_b}}{1+\frac{ds}{b\omega_h}}\right)^\alpha . \tag{8.28}$$

Thus, the fractional-order differentiator is defined as

$$s^\alpha \approx \left(\frac{d\omega_b}{b}\right)^\alpha \left(\frac{ds^2+b\omega_h s}{d(1-\alpha)s^2+b\omega_h s+d\alpha}\right) \left(\frac{1+\frac{bs}{d\omega_b}}{1+\frac{ds}{b\omega_h}}\right)^\alpha . \tag{8.29}$$

Expression (8.29) is stable if and only if all the poles are on the left-hand side of the complex s-plane. It is easy to check that expression (8.29) has three poles:

- One of the poles is located at $-b\omega_h/d$, which is a negative real pole since $\omega_h > 0$, $b > 0$, $d > 0$;
- The two other poles are the roots of the equation

$$d(1-\alpha)s^2 + a\omega_{hs} + d\alpha = 0 \tag{8.30}$$

  whose real parts are negative since $0 < \alpha < 1$.

Thus, all the poles of (8.29) are stable within the frequency range (ω_b, ω_h).

The irrational fractional-order part of expression (8.29) can be approximated by the continuous-time rational model

$$K(s) = \lim_{N\to\infty} K_N(s) = \lim_{N\to\infty} \prod_{k=-N}^{N} \frac{1+s/\omega'_k}{1+s/\omega_k}. \tag{8.31}$$

According to the recursive distribution of real zeros and poles, the zero and pole of rank k can be written as

$$\omega'_k = \left(\frac{d\omega_b}{b}\right)^{\frac{\alpha-2k}{2N+1}}, \quad \omega_k = \left(\frac{b\omega_h}{d}\right)^{\frac{\alpha+2k}{2N+1}}. \tag{8.32}$$

Thus, the continuous rational transfer function model can be obtained [107] as

$$s^\alpha \approx \left(\frac{d\omega_h}{b}\right)^\alpha \left(\frac{ds^2+b\omega_h s}{d(1-\alpha)s^2+b\omega_h s+d\alpha}\right) \prod_{k=-N}^{N} \frac{s+\omega'_k}{s+\omega_k}. \tag{8.33}$$

Through confirmation by experimentation and theoretical analysis, the synthesis approximation can obtain the good effect when $b = 10$ and $d = 9$.

Through the approximation method, the fractional-order system may be approximated as the very high integer-order system. The high integer-order rational transfer function could be very tedious.

With the above algorithm, a MATLAB function `new_fod()` is written

```
function G=new_fod(r,N,wb,wh,b,d)
if nargin==4, b=10; d=9; end
mu=wh/wb; k=-N:N; w_kp=(mu).^((k+N+0.5-0.5*r)/(2*N+1))*wb;
w_k=(mu).^((k+N+0.5+0.5*r)/(2*N+1))*wb; K=(d*wh/b)^r;
G=zpk(-w_kp',-w_k',K)*tf([d,b*wh,0],[d*(1-r),b*wh,d*r]);
```

with the syntax G_f=`new_fod(`γ,N,ω_b,ω_h,b,d`)`.

Example 8.4. Consider a model

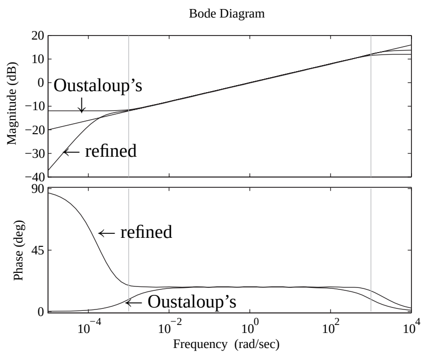
$$G(s)=\frac{s+1}{10s^{3.2}+185s^{2.5}+288s^{0.7}+1}$$

which is a fractional-order model. The exact Bode diagram can be obtained with the `bode()` function. The approximations using the Oustaloup filter, and the refined Oustaloup filter, can be obtained as shown in Figure 8.3(a). The approximations to the $G(s)$ model are shown in Figure 8.3(b). It can be seen that the refined method provides a much better fit:

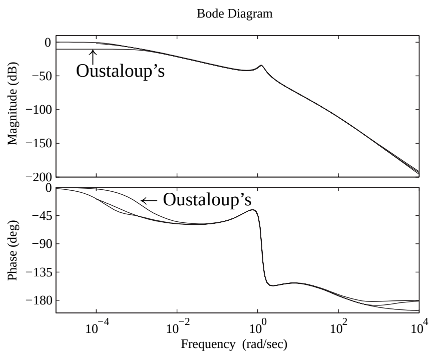
```
>> b=[1 1]; a=[10,185,288,1]; nb=[1 0]; na=[3.2,2.5,0.7,0];
   w=logspace(-4,4,200); G0=fotf(a,na,b,nb); H=bode(G0,w);
   s=zpk('s'); N=4; w1=1e-3; w2=1e3; b=10; d=9;
   g1=oustafod(0.2,N,w1,w2); g2=oustafod(0.5,N,w1,w2); a1=g1;
   g3=oustafod(0.7,N,w1,w2);
   G1=(s+1)/(10*s^3*g1+185*s^2*g2+288*g3+1);
   g1=new_fod(0.2,N,w1,w2,b,d); g2=new_fod(0.5,N,w1,w2,b,d);
   g3=new_fod(0.7,N,w1,w2,b,d); bode(g1,a1); figure
   G2=(s+1)/(10*s^3*g1+185*s^2*g2+288*g3+1); bode(H,G1,G2)
```

8.3.3 Simulink-Based Fractional-Order Nonlinear Differential Equation Solutions

From the previous discussions, it can be found that the refined Oustaloup recursive filter is an effective way to compute the fractional-order derivatives. It should be noted that the



(a) $s^{0.2}$ fittings



(b) Bode diagram comparisons

Figure 8.3. *Bode diagram comparisons.*

orders of the numerator and the denominator in the refined Oustaloup filter are the same, which may cause algebraic loops in Simulink. To avoid the algebraic loops, the filter should be followed by a low-pass filter, with a crossover frequency ω_h. The constructed block is shown in Figure 8.4(a).

With the mask facilities provided in Simulink, the fractional-order differentiator block can be built, as shown in Figure 8.4(b). Double click the fractional-order differentiator block to display the dialog box in Figure 8.4(c), which allows the user to enter parameters into the refined Oustaloup filters:

```
wb=ww(1); wh=ww(2); G=new_fod(gam,n,wb,wh,10,9);
num=G.num{1}; den=G.den{1}; T=1/wh; str='Fractional\n';
if isnumeric(gam)
   if gam>0, str=[str, 'Der  sˆ' num2str(gam) ];
   else, str=[str, 'Int  sˆ{' num2str(gam) '}']; end
else, str=[str, 'Der  sˆgam']; end
```

In practical simulation processes, the model established could be made up of stiff systems. Thus, ode15s or ode23tb algorithms should be selected to ensure high efficiency and accuracy. Examples will be given to demonstrate the solutions of FODEs.

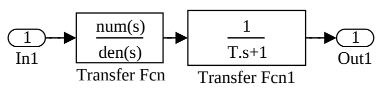
Example 8.5. Consider the nonlinear FODE described by

$$\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}+\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}+\frac{4}{3}y(t)=5\sin(10t).$$

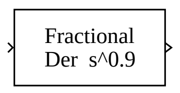
It can be seen that solving the original FODE is very complicated. From the original equation, the output signal $y(t)$ can explicitly be expressed as

$$y(t)=\frac{3}{4}\left[5\sin(10t)-\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}-\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}\right].$$

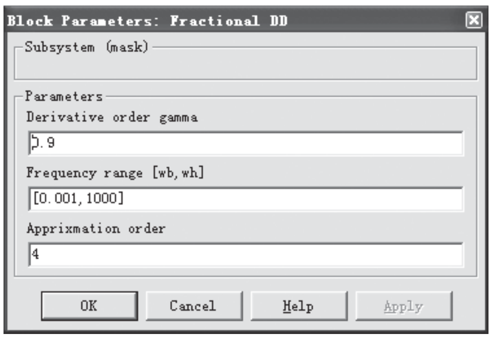
A Simulink model can then be established from the above equations, as shown in Figure 8.5(a). It can be seen from the model that each fractional-order differentiator can be modeled with the above designed block. In Figure 8.5(b), the simulation results are shown, with different parameters of the refined Oustaloup filter.



(a) fractional-order filter



(b) masked block (file: c7mfode.mdl)


Block Parameters: Fractional DD

Subsystem (mask)

Parameters

Derivative order gamma

0.9

Frequency range [wb,wh]

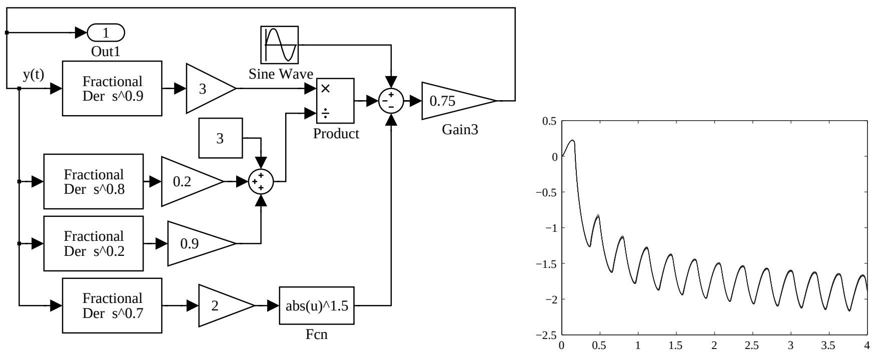
[0.001,1000]

Apprixmation order

4

(c) Dialog box of fractional-order differentiators

Figure 8.4. *Fractional-order differentiator block design.*



(a) Simulink model (file: c7mfod2.mdl) (b) simulation results

Figure 8.5. *Simulink modeling and results of a nonlinear FODE.*

It can be seen that the results are the same, and the only exception is the combination of $\omega_b = 0.001, \omega_h = 1000, N = 2$. However, even with this rough approximation, the error is still acceptable.

8.4 Model Reduction Techniques for Fractional-Order Systems

It has been shown that if the integer-order approximation is used to fit the fractional-order transfer function models with the use of the refined Oustaloup recursive filter, the order of the final system could be extremely high. Thus, a low-order approximation to the original problem can be found using the optimal model reduction method.

Recall the expected reduced-order model given by

$$G_{r/m,\tau}(s) = \frac{\beta_1 s^r + \cdots + \beta_r s + \beta_{r+1}}{s^m + \alpha_1 s^{m-1} + \cdots + \alpha_{m-1}s + \alpha_m} \mathrm{e}^{-\tau s}. \tag{8.34}$$

An objective function for minimizing the $\mathcal{H}_2$-norm of the reduction error signal $e(t)$ can be defined as

$$J = \min_{\theta} \left\| \widehat{G}(s) - G_{r/m,\tau}(s) \right\|_2, \tag{8.35}$$

where θ is the set of parameters to be optimized such that

$$\theta = [\beta_1, \ldots, \beta_r, \alpha_1, \ldots, \alpha_m, \tau]. \tag{8.36}$$

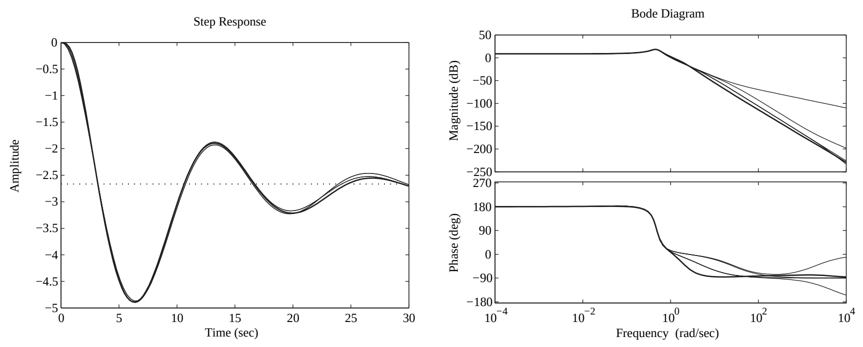
For an easy evaluation of the criterion J, the delayed term in the reduced-order model $G_{r/m,\tau}(s)$ can be further approximated by a rational function $\widehat{G}_{r/m}(s)$ using the Padé approximation technique [47]. Thus, the revised criterion can then be defined by

$$J = \min_{\theta} \left\| \widehat{G}(s) - \widehat{G}_{r/m}(s) \right\|_2 \tag{8.37}$$

and the $\mathcal{H}_2$-norm computation can be evaluated recursively using the algorithm in [108]. The function `opt_app()` discussed in Sec. 3.6 can still be used for fractional-order systems.

Table 8.1. *Comparisons of different order combinations.*

r	m	Reduced-order model	Error
2	3	$\dfrac{0.03147s^2 - 0.8141s - 0.07206}{s^3 + 0.3168s^2 + 0.2582s + 0.02703}$	0.2286
2	4	$\dfrac{-0.0119s^2 - 23.21s - 2.035}{s^4 + 28.78s^3 + 9.242s^2 + 7.365s + 0.7634}$	0.2308
2	5	$\dfrac{-4.932s^2 - 0.8602s - 0.00386}{s^5 + 5.741s^4 + 2.794s^3 + 1.596s^2 + 0.3134s + 0.001448}$	0.1342
2	6	$\dfrac{-2.327\times10^4s^2 - 4059s - 18.21}{s^6+4719s^5+2.709\times10^4s^4+1.318\times10^4s^3+7534s^2+1479s+6.831}$	0.1342



(a) step responses (b) Bode diagrams

Figure 8.6. *Comparisons of the reduced-order models.*

Example 8.6. Consider again the high-order fractional-order transfer function given in Example 8.3, where a 48th-order model was obtained, and with the refined Oustaloup filter, a 58th-order model can be obtained. Using optimal reduction techniques for different order combinations, the reduced-order models can be found as shown in Table 8.1. It can be seen that the $G_{2/5}(s)$ model is the best one. The step responses and Bode diagrams are compared in Figure 8.6. It can be seen that the approximation is satisfactory. It should be noted that in the code, the `opt_app()` function may be called several times since the original model should be used in these cases.

```
>> N=4; w1=1e-3; w2=1e3; s=tf('s'); g1=new_fod(0.501,N,w1,w2,9,10);
   g2=new_fod(0.42,N,w1,w2,9,10); g3=new_fod(0.798,N,w1,w2,9,10);
   g4=new_fod(0.31,N,w1,w2,9,10); g5=new_fod(0.63,N,w1,w2,9,10);
   G=(-2*g5-4)/(2*s^3*g1+3.8*s^2*g2+2.6*s*g3+2.5*s*g4+1.5);
   Gr1=opt_app(G,2,3,0);norm(G-Gr1),Gr2=opt_app(G,2,4,0);norm(G-Gr2)
   Gr3=opt_app(G,2,5,0); Gr3=opt_app(G,2,5,0,Gr3); norm(G-Gr3)
   Gr4=opt_app(G,2,6,0); Gr4=opt_app(G,2,6,0,Gr4);
   Gr4=opt_app(G,2,6,0,Gr4); norm(G-Gr4)
   step(G,Gr1,Gr2,Gr3,Gr4,30); figure; bode(G,Gr1,Gr2,Gr3,Gr4)
```

8.5 Controller Design Studies for Fractional-Order Systems

From the analysis given previously, it can be seen that the behaviors of fractional-order controllers may be different from their integer-order counterparts. For instance, if the widely used PID controller is considered, its fractional-order version PI$^\lambda$D$^\mu$ controller can be expressed by [99]

$$G_c(s) = K_p + \frac{K_i}{s^\lambda} + K_d s^\mu. \tag{8.38}$$

In the illustration in Figure 8.7, the fractional-order PID controller is explained, with the horizontal axis as the order of the integrator and the vertical axis the order of the differentiator. It can be seen that the ordinary PI (proportional plus integral), PD, and PID controllers are special cases of the fractional-order PID controller since the values of λ and μ can be selected freely, which adds two more degree of freedom to the controller design. It has been shown that the control behavior of the best fractional-order PID controller is quite superior to the best conventional PID controller in some applications [109].

If the loop shaping technique is considered, it can be seen that the Bode magnitude diagrams is no longer restricted to $20k$ dB/decade slopes. Thus the shape of the loop transfer function can be set freely for better performance and robustness. In this section, several examples will be given to show the design of an integer-order controller and fractional-order controller for fractional-order plants.

Example 8.7. For a plant model

$$G(s) = \frac{1}{s^{2.6} + 2.2s^{1.5} + 2.9s^{1.3} + 3.32s^{0.9} + 1},$$

if an integer-order PID controller is expected, it is quite natural to first find an FOPDT approximate model,

$$G_p(s) = k\frac{\mathrm{e}^{-Ls}}{Ts+1}$$

and then design a PID controller for the FOPDT model. The designed controller can then be used in closed-loop control of the fractional-order plant $G(s)$. For instance, the Wang–Juang–Chan algorithm [69] in Sec. 6.3.4 can be used to design a PID controller for an

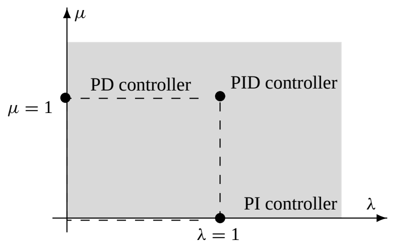


Figure 8.7. *Fractional-order PID controller.*

FOPDT model with an optimum ITAE criterion:

$$K_p = \frac{(0.7303 + 0.5307T/L)(T + 0.5L)}{K(T + L)}, \quad T_i = T + 0.5L, \quad T_d = \frac{0.5LT}{T + 0.5L}. \tag{8.39}$$

The following statements can be used to extract the FOPDT model from the approximated high-order plant model:

```
>> N=4; w1=1e-3; w2=1e3; s=tf('s');
   g1=new_fod(0.6,N,w1,w2,9,10); g2=new_fod(0.5,N,w1,w2,9,10);
   g3=new_fod(0.3,N,w1,w2,9,10); g4=new_fod(0.9,N,w1,w2,9,10);
   G=1/(s^2*g1+2.2*s*g2+2.9*s*g3+3.32*g4+1); Gr=opt_app(G,0,1,1)
```

The reduced plant model is then

$$G_r(s) = \frac{0.1702}{s + 0.1702} \mathrm{e}^{-0.612s}.$$

The PID controller can be designed such that

```
>> K=0.1702/0.1702; T=1/0.1702; L=0.612;
   Ti=T+0.5*L; Kp=(0.7303+0.5307*T/L)*Ti/(K*(T+L));
   Td=(0.5*L*T)/(T+0.5*L); Gc=Kp*(1+1/Ti/s+Td*s),
```

The integer-order PID controller is designed as

$$G_c(s) = 4.7960 \left(1 + \frac{1}{5.6315s} + 0.3076s \right) = \frac{1.614s^2 + 5.55s + 0.8979}{s}.$$

Under such a controller, the closed-loop step response is obtained as shown in Figure 8.8. It can be seen that the integer-order PID controller can still be used in the fractional-order plant control. The control results are satisfactory. It is also seen that the high-order approximation to the closed-loop system is very accurate:

```
>> Gcf=fotf(1,1,[1.614 5.55 0.8979],[2,1,0]); H=fotf(1,0,1,0);
   a=[1 2.2 2.9 3.32 1]; an=[2.6,1.5,1.3 0.9 0]; G0=fotf(a,an,1,0);
   GG=feedback(Gcf*G0,H); t=0:0.005:15;
   step(feedback(G*Gc,1),t); hold on, step(feedback(G0*Gcf,H),t);
```

Example 8.8. Consider a fractional-order plant model

$$G(s) = \frac{10}{s^{\alpha} + 2.2},$$

where the order α is an undetermined parameter, within the interval $\alpha \in (1.2, 1.6)$. The nominal value of the variable is $\alpha_0 = 1.4$. In order to get a low-order robust controller, a relatively smaller value of N can be selected, for instance, $N = 2$. The following statements can be used to approximate the original model by integer-order approximation such that

```
>> N=2; w1=1e-3; w2=1e3; s=tf('s');
   g1=oustafod(0.4,N,w1,w2);  G=1/(s*g1+2.2);
```

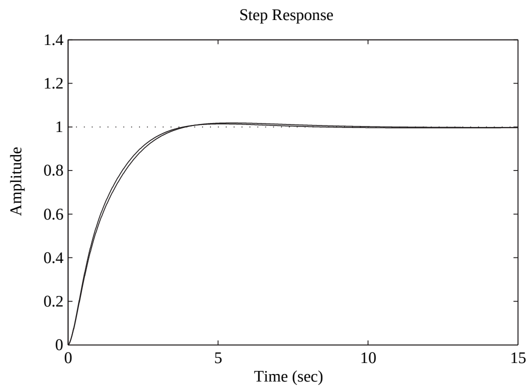


Figure 8.8. *Integer-order PID control of fractional-order plant.*

Select weighting functions $w_1(s) = 100/(s+1)$ and $w_3(s) = 10/(0.01s+100)$. The optimal $\mathscr{H}_\infty$ controller can be designed such that

```
>> W1=100/(s+1); W3=100/(0.01*s+100); Gc=mixsyn(G,W1,[],W3);
```

The controller can be designed as

$$G_c(s) = \frac{\begin{array}{c}71870205(s+1000)(s+144.3)(s+8.265)(s+0.1116)\\(s+0.006921)(s^2+1.73s+2.388)\end{array}}{\begin{array}{c}(s+9499)(s+9975)(s+346.4)(s+27.46)\\(s+1.738)(s+1)(s+0.1096)(s+0.006918)\end{array}}.$$

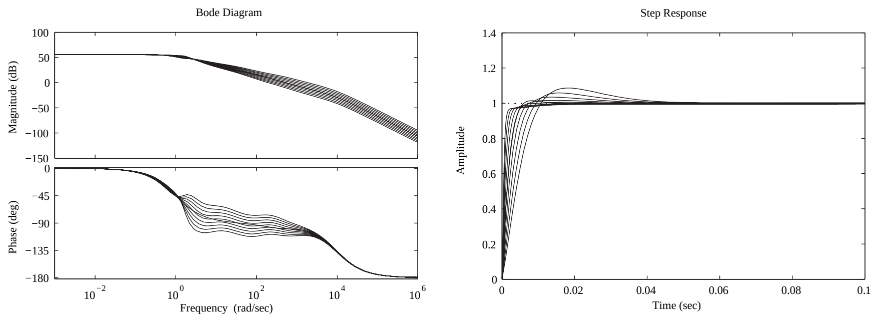
Under such a controller, the open-loop Bode diagrams and the closed-loop step response are obtained as shown in Figures 8.9(a) and (b), respectively:

```
>> f1=figure; bode(G*Gc); hold on
   f2=figure; step(feedback(G*Gc,1),0.1); hold on
   for a=[0.2:0.05:0.6]
      g1=oustafod(a,4,w1,w2); G1=1/(s*g1+2.2);
      figure(f1); bode(G1*Gc);
      figure(f2); step(feedback(G1*Gc,1),0.1)
   end
```

Example 8.9. Consider again the fractional-order plant model in Example 8.7. The integer-order approximation can be obtained such that

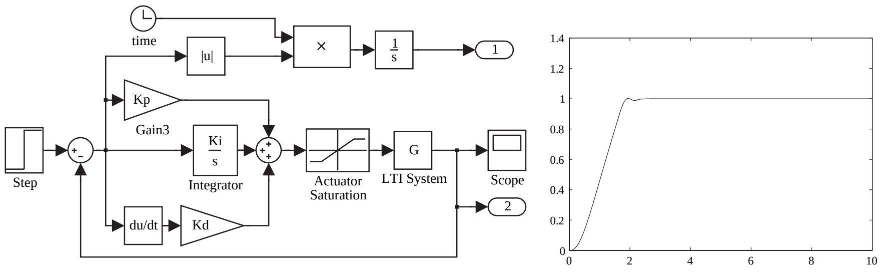
```
>> N=4; w1=1e-3; w2=1000; s=tf('s');
   g1=oustafod(0.6,N,w1,w2); g2=oustafod(0.5,N,w1,w2);
   g3=oustafod(0.3,N,w1,w2); g4=oustafod(0.9,N,w1,w2);
   G=1/(s^2*g1+2.2*s*g2+2.9*s*g3+3.32*g4+1);
```

Using the integer-order model, the Simulink model for optimal controller design with an integer-order PID controller is established as shown in Figure 8.10(a). A saturation actuator with limits ±5 is also included in the Simulink model.



(a) Bode diagrams (b) closed-loop step responses

Figure 8.9. *Time and frequency domain analysis under robust controller.*



(a) Simulink model (file: c8mfpid2.mdl) (b) closed-loop response

Figure 8.10. *Optimal PID controller design for fractional-order plant.*

It can be found by using the Optimal Controller Designer (OCD) program that the parameters of the PID controller are $K_p = 14768.1007$, $K_i = 1.35636077$, $K_d = 2306.39271$. Under such a controller, the optimum step response of the closed-loop system can be obtained as shown in Figure 8.10(b). It can be seen that the controller obtained with the OCD is much better than the one obtained in Example 8.7. Also the control action is restricted within the specific range.

Due to the robustness of the PID controllers, the errors in the controller parameters may not cause any problem in the control results. For instance, if we had the erroneous parameters $K_p = 10000$, $K_i = 1$, $K_d = 2500$, where the errors reach 35%, the control results would be as shown in Figure 8.11(a). It can be seen that the system responses are almost the same with the optimal PID controller:

```
>> Kp=10000; Ki=1; Kd=2500;
   [t,x,y]=sim('c8mfpid2',[0,10]); plot(t,y(:,2))
```

Assume that plant model is changed to

$$G(s) = \frac{2}{s^{2.6} + 5s^{1.5} + 4s^{1.3} + 5.32s^{0.9} + 1},$$

where the parameters are all perturbed. If the erroneous PID controller is still used, the control results are as shown in Figure 8.11(b). It can be seen that, although the plant models

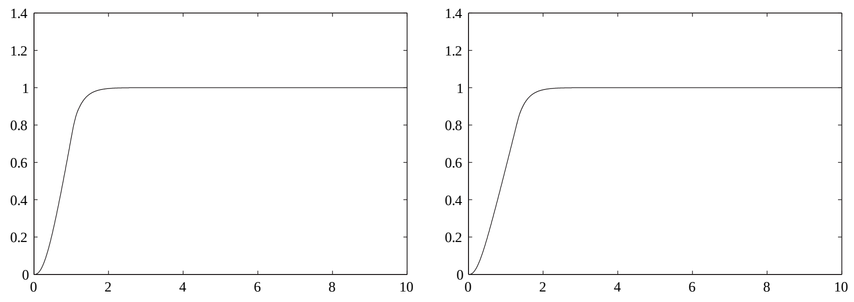

(a) controller parameters change (b) plant and controller change

Figure 8.11. *The robustness of the PID controller.*

change significantly, the PID controller can still behave perfectly. This demonstrates the robustness of the PID controller in fractional-order plant models:

```
>> G=2/(s^2*g1+5*s*g2+4*s*g3+5.32*g4+1);
   [t,x,y]=sim('c8mfpid2',[0,10]); plot(t,y(:,2))
```

Problems

1. Assume that a fractional-order linear differential equation is given by

$$0.8\mathscr{D}_t^{2.2}y(t) + 0.5\mathscr{D}_t^{0.9}y(t) + y(t) = 1,$$

with initial values $y(0) = y'(0) = y''(0) = 0$. Solve numerically the FODE. If the order of 2.2 is approximated by 2, and 0.9 is approximated by 1, the original fractional-order differential equation can be approximated by an integer-order system. Compare the accuracy of the approximated integer-order systems.

2. For a fractional-order model given by

$$(a). \quad G(s) = \frac{5}{s^{2.3} + 1.3s^{0.9} + 1.25}$$

and

$$(b). \quad G(s) = \frac{5s^{0.6} + 2}{s^{3.3} + 3.1s^{2.6} + 2.89s^{1.9} + 2.5s^{1.4} + 1.2},$$

approximate the fractional-order models with low-order integer-order models, and compare the accuracy of the frequency and time domain fittings. Discuss what order combination is most suitable for the original model.

3. Suppose that the plant model is

$$G(s) = \frac{1}{s^{2.6} + 2.2s^{1.5} + 2.9s^{1.3} + 3.32s^{0.9} + 1},$$

and an integer-order PID controller is

$$G_c(s) = \frac{1.614s^2 + 5.55s + 0.8979}{s}.$$

Find the closed-loop fractional-order model.

4. Write a function to find the solutions to the FODE using the algorithm in (8.17)–(8.21), and compare the results with the Grünwald–Letnikov definition approach and the block diagram algorithm.

5. Consider the linear FODE given by

$$\mathscr{D}x(t) + \left(\frac{9}{1+2\lambda}\right)^{\alpha} \mathscr{D}^{\alpha}x(t) + x(t) = 1,$$

where $\lambda = 0.5$, $\alpha = 0.25$ and $x(0) = 0$. Solve the equation numerically.

6. Find a good approximation to $s^{0.7}$ with the revised Oustaloup filter and see which N can best fit the fractional-order differentiator.

7. Solve the following nonlinear FODE with the block diagram algorithm with $x(0) = 0$:

$$\mathscr{D}^2x(t) + \mathscr{D}^{1.455}x(t) + \left[\mathscr{D}^{0.555}x(t)\right]^2 + x^3(t) = \sin t.$$

8. For the plant model

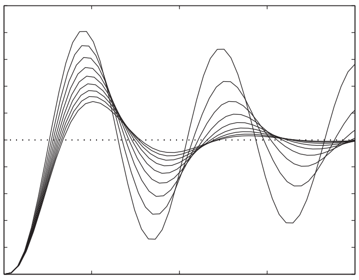
$$G(s) = \frac{5s^{0.6} + 2}{s^{3.3} + 3.1s^{2.6} + 2.89s^{1.9} + 2.5s^{1.4} + 1.2},$$

design an integer-order PID controller and observe the control results.

9. For the fractional-order model

$$G = \frac{b}{as^{0.7} + 1},$$

design an $\mathscr{H}_\infty$ controller which can tolerate the parameter changes in the fractional-order model, for instance, $a \in (0.2, 5)$ and $b \in (0.2, 1.5)$.



Appendix

CtrlLAB: A Feedback Control System Analysis and Design Tool

A.1 Introduction

A.1.1 What Is CtrlLAB?

CtrlLAB, a MATLAB-based toolkit with an integrated graphical user interface (GUI), was designed by the authors for solving the modeling, analysis, and design problems in SISO (single input–single output) feedback control systems. It is developed from the old Control Kit by the authors [110]. CtrlLAB has become a flexible and powerful tool for both teaching and engineering design and requires minimum user effort. It can be used as a companion to this book.

CtrlLAB, written and tested under MATLAB v4.2, was first made public on the MathWorks anonymous ftp site as a user-contributed MATLAB program. Since then, much useful feedback has been received. Over the years, CtrlLAB has been greatly improved. It has already been used as a CAI (computer aided instruction) tool in control courses at many universities worldwide. The latest version of CtrlLAB can also run under other versions of MATLAB, including MATLAB R2007b. It is still freely downloadable from MATLAB Central at

```
http://www.mathworks.com/matlabcentral/index.shtml
```

Currently, CtrlLAB is the most downloaded tool under the Controls and Systems Modeling file exchange category at MATLAB Central.

The main facilities provided by CtrlLAB are

- model entry, including Simulink model entry;
- model display;
- state space realizations;
- model reduction using various algorithms;
- system analysis in frequency and time domains;
- graphical display with figure editing and manipulation;
- a GUI matrix processor and editor;

- many controller design modules such as the model-based approaches (lead-lag, LQ (linear quadratic) optimal, pole-placement, etc.); PID (proportional integral derivative) parameter setting and PID tuning schemes; and robust controller design approaches (such as LQG (linear quadratic Gaussian), LQG/LTR (loop transfer recovery), $\mathcal{H}_2$, $\mathcal{H}_\infty$, etc.).

A.1.2 Installation and Requirements

With the downloaded `ctrllab.zip` file, unzip it to a directory using `WinZip` or `pkunzip` software. Before running CtrlLAB, the directory of CtrlLAB should be added to the MATLAB path. This can be set with the File | Set Path menu item in the MATLAB command window.

CtrlLAB is written for the PC Windows platform; however, it should also be able to run on other platforms. Although CtrlLAB has not been fully tested on other platforms, with a MATLAB version newer than 4.2c, the cross platform compatibility will be much better than what was experienced under MATLAB version 4.2c. We believe that CtrlLAB can run on any current version of other platforms with little modification.

A.1.3 Execution of CtrlLAB

To run CtrlLAB, simply type `ctrllab` under the MATLAB prompt, and a GUI with menus will pop up, as shown in Figure A.1. The user must first enter or to define the models, which include the plant, the controller, and the feedback element. The default models for the latter two are all unity. The possible time delay may also be specified. With the specified models, the analysis and design tasks can be performed.

Menus and dialog boxes are provided to invoke relevant functions to fulfill the user's own analysis and design tasks. Note that all the functions provided in CtrlLAB can be accessed through the efficient and user friendly GUI. There is no need to call these functions

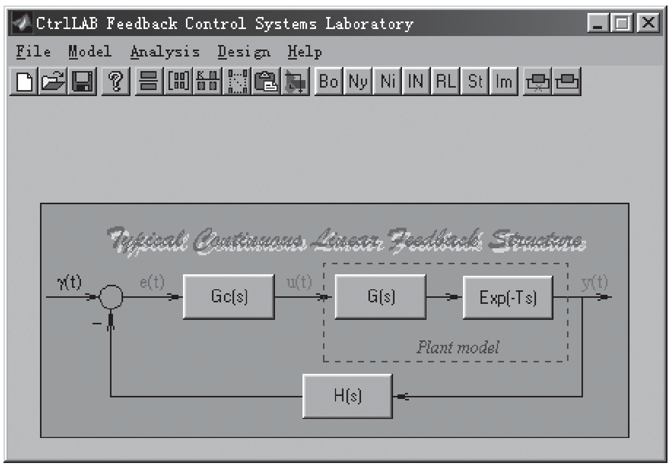

CtrlLAB Feedback Control Systems Laboratory
File Model Analysis Design Help
Bo Ny Ni IN RL St Im
Typical Continuous Linear Feedback Structure
γ(t)
e(t)
Gc(s)
u(t)
G(s)
Exp(-Ts)
y(t)
Plant model
H(s)
–

Figure A.1. *The GUI of CtrlLAB.*

manually. CtrlLAB is designed for linear feedback control system analysis and design using only mouse clicks and numeric key strokes. Great effort has been made in CtrlLAB to minimize the user involvement in the analysis and design of feedback control systems.

A.2 Model Entry and Model Conversion

A.2.1 Transfer Function Entry

To quickly enter a default model, the user can click one of the model icons in the block diagram shown in Figure A.1, and CtrlLAB will check whether the model exists in the work space. If it does not exist, a dialog box, shown in Figure A.2, will appear by default, which allows the user to enter the system model by specifying the numerator and denominator, respectively, in the appropriate edit boxes.

The transfer function model can be entered in two ways. The first is by entering the standard MATLAB vectors in descending order of the Laplace complex variable s. The second is by representing the polynomials in a "natural way." These two methods are demonstrated in Table A.1. It can be seen that for the factorized polynomials, the s polynomial representation is much more "natural" and simpler than a pure MATLAB expression.

A.2.2 Entering Other Model Representations

The state space model, or zero-pole-gain model, can also be entered if the corresponding item from the list box shown in Figure A.2 is selected.

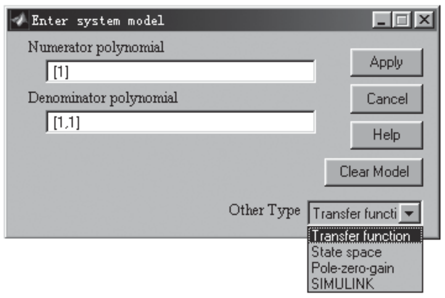
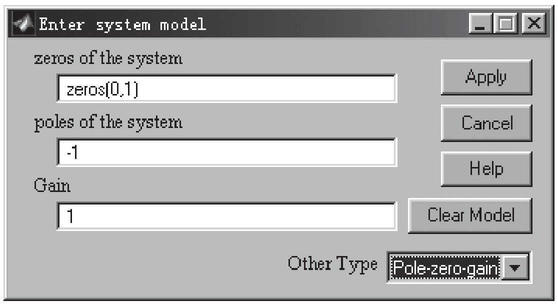

Enter system model
Numerator polynomial
[1]
Denominator polynomial
[1,1]
Other Type: Transfer functi

Figure A.2. *Dialog box for transfer function model entry.*

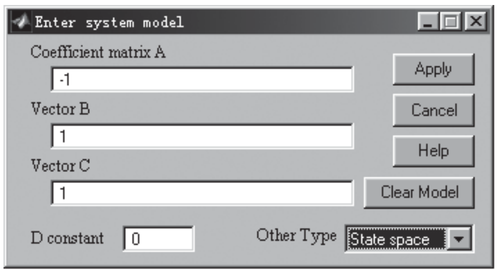
Table A.1. *Examples of polynomial representations.*

Mathematical	MATLAB commands	s polynomial
$s^2 + 5s + 4$	`[1,5,4]`	`s2+5s+4`
$s^2(s + 5)(s^2 + 7)$	`[conv([1,5],[1,0,7]),0,0]`	`s2(s+5)(s2+7)2`
$1.5s^3(s^3+7s^2+6s+2)^{12}$	too complicated	`1.5s3(s3+7s2+6s+2)12`



zeros of the system: zeros(0,1)
poles of the system: -1
Gain: 1
Other Type: Pole-zero-gain

Figure A.3. *Dialog box for zero-pole-gain model entry.*



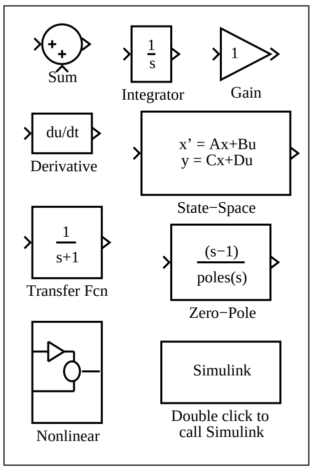
Coefficient matrix A: -1
Vector B: 1
Vector C: 1
D constant: 0
Other Type: State space

Figure A.4. *Dialog box for state space model entry.*

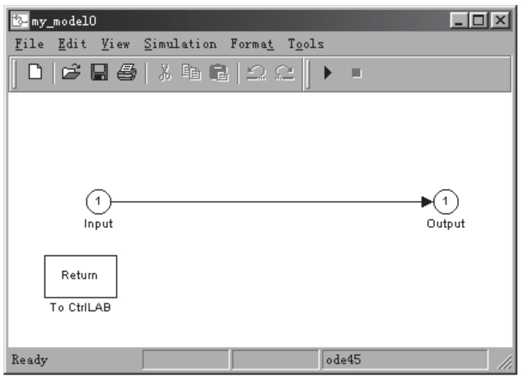
In Figure A.2, if the menu item pole-zero (for zero-pole-gain model) is selected, the dialog box shown in Figure A.3 will appear, where the zero-pole-gain model parameters can be entered in the corresponding edit boxes. Then, press the OK button to confirm. Internally, a transfer function object will be generated automatically from the user-specified zero-pole-gain model. For the state space item, the dialog box shown in Figure A.4 will appear, where the $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, D)$ matrices of the system can be entered in the corresponding edit boxes. Then, a transfer function object of the block can be generated automatically from the given state space model.

A.2.3 A More Complicated Model Entry

If the system model under study has a more complicated structure, such as containing complex block diagrams or nonlinearities, the Simulink program should be used to construct the system model. In this case, the user can select the Simulink item from the dialog box shown in Figure A.2. A model name (an internal name) will be requested and then the Simulink editing environment will appear, as shown in Figures A.5(a) and (b), where Figure A.5(a) is the model library from which all the Simulink library models can be accessed. Figure A.5(b) is a blank Simulink model editing window in which the user can draw the system model between the input and output ports of the system. Once the model entry process is completed in the Simulink edit window, as shown in Figure A.5(b), double click Return to CtrlLAB to return the user system model to CtrlLAB. If the user model in Simulink is nonlinear, the linearized transfer function model of the user system will be created and saved, together with the original Simulink model, for CtrlLAB use. A simple nonlinear model entry example in



(a) model library



(b) model entering window

Figure A.5. *Simulink model entering in CtrlLAB.*

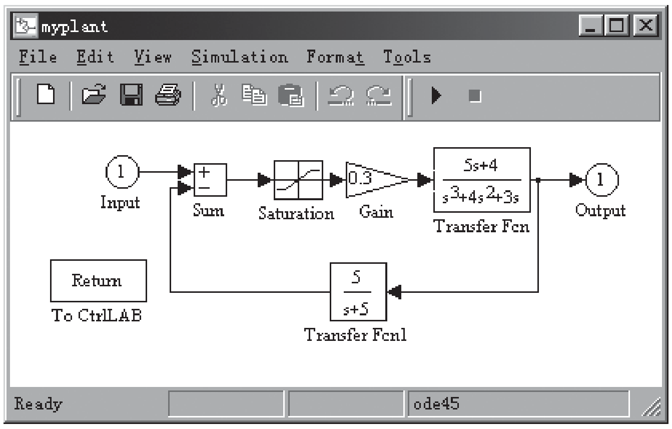


Figure A.6. *Complicated model entry in CtrlLAB via Simulink.*

CtrlLAB is shown in Figure A.6 which uses Simulink to describe the nonlinear part. Note the Return to CtrlLAB button in Figure A.6 for returning a linearized transfer function object for use with CtrlLAB.

A.3 Model Transformation and Reduction

A.3.1 Model Display

To display the model of a block in Figure A.1, select Model | Model Select in the menu shown in Figure A.7, or simply click the relevant block button in the main interface shown in Figure A.1.

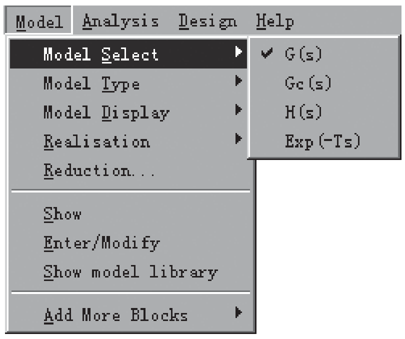
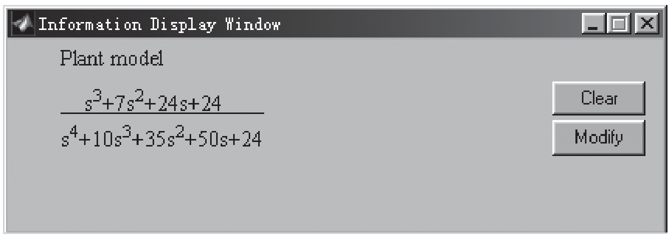


Figure A.7. *Model selecting menu.*



Plant model

$$\frac{s^3+7s^2+24s+24}{s^4+10s^3+35s^2+50s+24}$$

Figure A.8. *Transfer function display.*

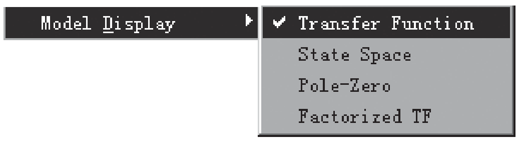


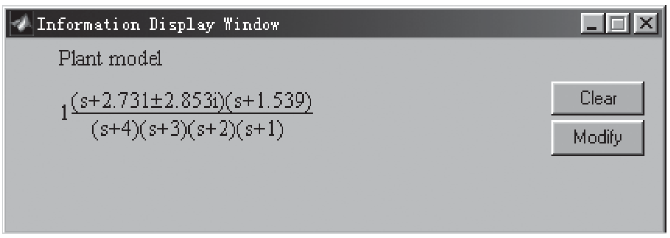
Figure A.9. *Display format selection.*

As an example, consider the transfer function of the plant model given by

$$G(s) = \frac{s^3 + 7s^2 + 24s + 24}{s^4 + 10s^3 + 35s^2 + 50s + 24}.$$

To display the transfer function model of the plant, simply press the G(s) button in the main interface shown in Figure A.1. The transfer function model will then be displayed in the Information Display Window as shown in Figure A.8. The displayed model can also be modified in the display window by pressing the Modify button. The dialog box shown in Figure A.2 will be displayed again for model parameter changes.

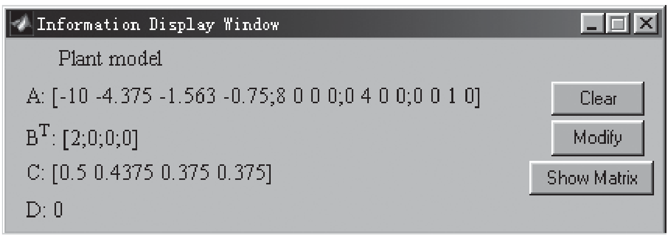
The block model can be displayed in various formats. This can be done by selecting the Model | Model Display menu, shown in Figure A.9, with the transfer function format as the default. Through the Model | Model Display | Factorized TF menu item, the transfer function in the factorized format will be displayed as shown in Figure A.10.



Plant model

$$1\frac{(s+2.731\pm 2.853i)(s+1.539)}{(s+4)(s+3)(s+2)(s+1)}$$

Figure A.10. *Factorized transfer function display format.*



Plant model

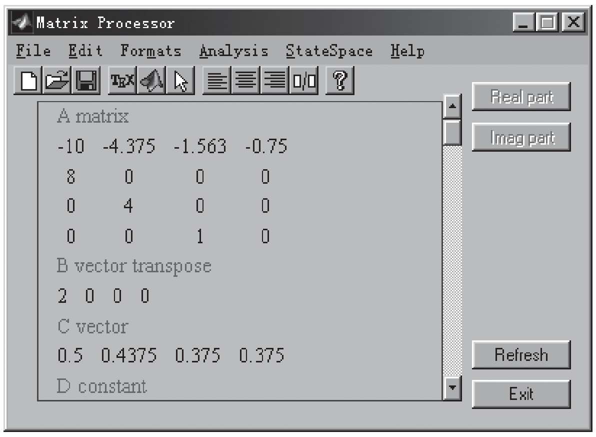
A: [-10 -4.375 -1.563 -0.75;8 0 0 0;0 4 0 0;0 0 1 0]

B^T: [2;0;0;0]

C: [0.5 0.4375 0.375 0.375]

D: 0

Figure A.11. *State space model display format.*

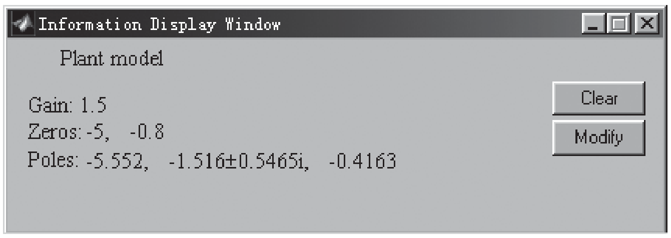


```
A matrix
-10   -4.375   -1.563   -0.75
 8      0        0        0
 0      4        0        0
 0      0        1        0
B vector transpose
2   0   0   0
C vector
0.5   0.4375   0.375   0.375
D constant
```

Figure A.12. *Display via the Matrix Processor.*

Moreover, the state space model can be displayed by the Model | Model Display | state space menu item as demonstrated in Figure A.11. When the Show button is clicked, the Matrix Processor is activated; the typical window is shown in Figure A.12. The zero-pole-gain format of the system is displayed by the Model | Model Display | Pole-Zero menu item which is shown in Figure A.13.

If the nonlinear system model is involved, only the linearized model will be displayed as in Figure A.14. To display the original Simulink model, simply press the to CtrlLAB button.

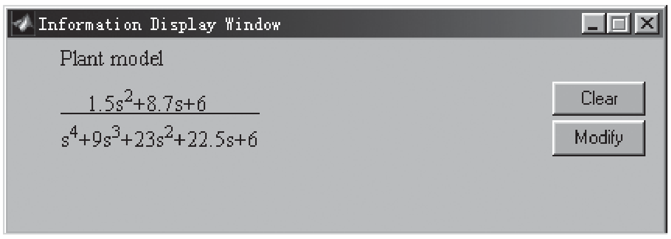


Information Display Window

Plant model

Gain: 1.5
Zeros: -5, -0.8
Poles: -5.552, -1.516±0.5465i, -0.4163

Figure A.13. *Zero-pole-gain display format.*



Information Display Window

Plant model

$$\frac{1.5s^2+8.7s+6}{s^4+9s^3+23s^2+22.5s+6}$$

Figure A.14. *Linearized model display.*

A.3.2 State Space Realizations

Different state space realizations can be performed for a given transfer function plant model. This can be done by the Model | Realisation menu items shown in Figure A.15, and an example of the Jordanian canonical form of the system is obtained, as shown in Figure A.16, via the Matrix Processor interface.

A.3.3 Model Reduction

Reduced-order models of the system can also be obtained by the Model | Reduction menu item. The model reduction dialog box will appear as in Figure A.17, where various model reduction approaches are implemented such as the continued-fraction approach, the Padé method, the Routh method, the dominant mode method, the balanced realization method, the optimal reduction method, the FF-Padé method, the modal method, and the optimal Hankel approximation method.

For example, if the Padé approximation method is chosen from the list box of model reduction methods, the expected order of the reduced model can be specified as in Figure A.17. The reduced-order model is then obtained as shown in Figure A.18.

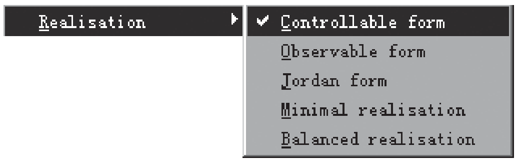
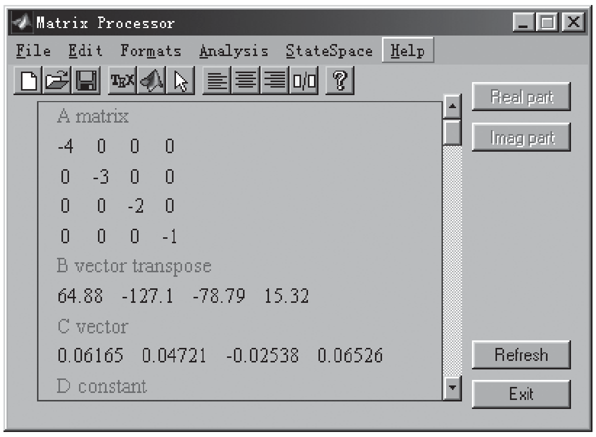
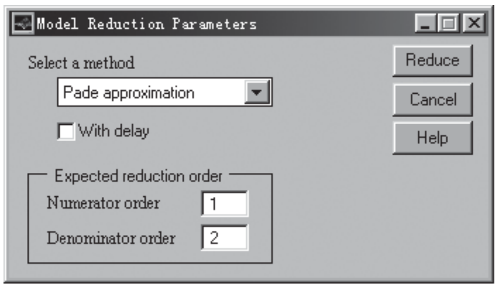


Figure A.15. *State space realization menu.*



```
A matrix
-4   0   0   0
 0  -3   0   0
 0   0  -2   0
 0   0   0  -1
B vector transpose
64.88   -127.1   -78.79   15.32
C vector
0.06165   0.04721   -0.02538   0.06526
D constant
```

Figure A.16. *Jordan realization.*



Select a method: Pade approximation

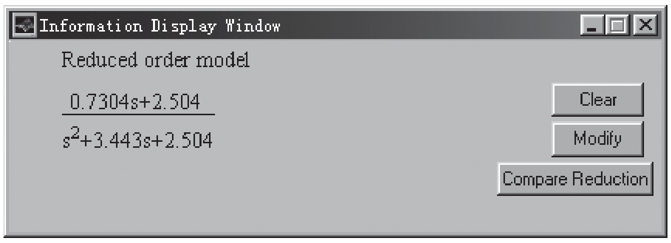
☐ With delay

Expected reduction order

Numerator order: 1

Denominator order: 2

Figure A.17. *Model reduction dialog box.*

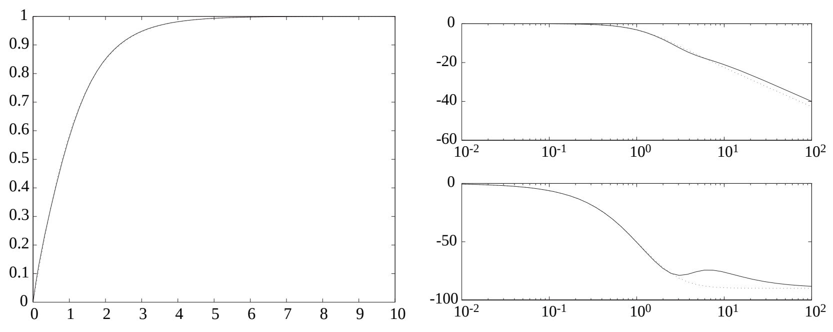


Reduced order model

$$\frac{0.7304s+2.504}{s^2+3.443s+2.504}$$

Figure A.18. *Model reduction result via the Padé approximation method.*

To compare the reduced-order model with the original model, click on Compare responses in the model display window. A new dialog box pops up for choosing a comparison plot from a list of responses which include the Bode diagrams, Nyquist plots, Nichols charts, as well as the step and impulse responses between the original model and the reduced-order model. For instance, the step response comparison, and the Bode diagram comparison, of the original system and the reduced model via the Padé approximation method are shown in Figures A.19(a) and (b), respectively, where the solid line represents for the original model and the dotted line the reduced-order model. It can be seen that the responses of the two



(a) step response comparison (b) Bode diagram comparison

Figure A.19. *Comparisons of the reduced order and the original models.*

models are quite close, especially in the step response comparison, where the two curves are almost indistinguishable.

A.4 Feedback Control System Analysis

Various linear system analysis tasks covered in this book can be performed by the direct use of CtrlLAB. After performing the model entry from Sec. A.2, select Analysis from the main menu shown in Figure A.1. The system analysis menu will appear as shown in Figure A.20. In this menu, plots for time domain, frequency domain, and root locus analysis can be generated by just using mouse clicks. In what follows, some detailed instructions are given in the subsections to follow.

A.4.1 Frequency Domain Analysis

The Bode diagram of the system can be obtained by the Analysis | Frequency Domain Analysis | Bode Diagram menu item. The result is shown as in Figure A.21(a).

Via the Options | Show asymptotes sub-menu in the Bode diagram window, the Bode plot asymptotes are drawn together with the exact Bode diagram, as demonstrated in Figure A.21(b).

The properties of the graphs can be modified by the Options | Plot preference sub-menu in the Bode diagram window, and a dialog box is then provided as shown in

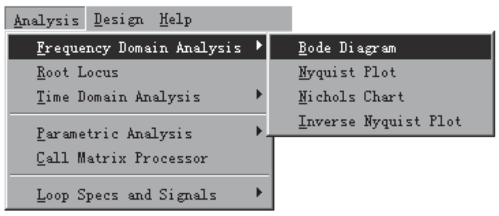
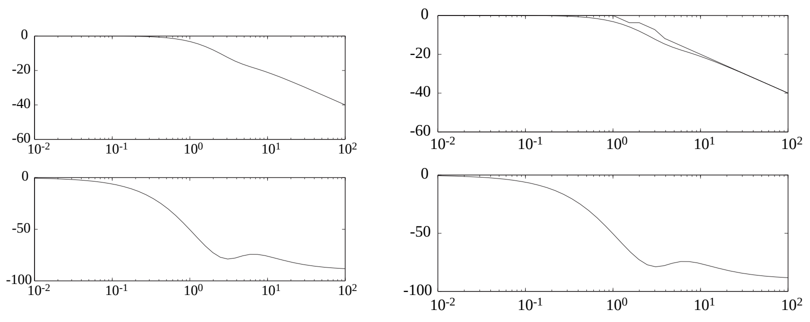
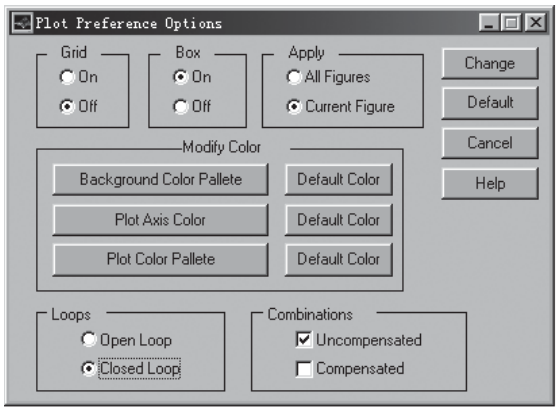


Figure A.20. *System analysis menu in CtrlLAB.*



(a) Bode diagram (b) with asymptotes

Figure A.21. *Bode diagram of a given linear system.*


Plot Preference Options

Grid: On [✓ Off]

Box: [✓ On] Off

Apply: All Figures [✓ Current Figure]

Modify Color: Background Color Pallete, Default Color; Plot Axis Color, Default Color; Plot Color Pallete, Default Color

Loops: Open Loop [✓ Closed Loop]

Combinations: [✓ Uncompensated] Compensated

Figure A.22. *Graph properties setting dialog box.*

Figure A.22, where some of the details on the graph can be modified such as the boxes, grid, colors, etc. Moreover, the open-loop and closed-loop properties of the plots can also be changed. If a controller model is available, the Combinations group can be used to choose the Compensated as well as the Uncompensated frequency response. For instance, if the user checks the Closed Loop box, the closed-loop Bode diagram can then be obtained as shown in Figure A.23.

The Nyquist and Nichols charts can be obtained via the Analysis | Nyquist Plot and Analysis | Nichols Chart menu items. Results shown in Figures A.24(a) and (b), respectively.

The root locus plot can be obtained by using Analysis | Root Locus. For some particular systems, the directly obtained root locus of the system may not be very informative due to the poor quality of the automatically chosen plot ranges. In this case, the user can change the axis of the plot via the Options | Zoom | User Define menu item on the root locus window. A dialog box then appears as shown in Figure A.25(a). The ranges of the x and y axes can be changed until a good display result is obtained. For instance, with the properly chosen axes, the more informative root locus of the system can then be redrawn, as shown in Figure A.25(b).

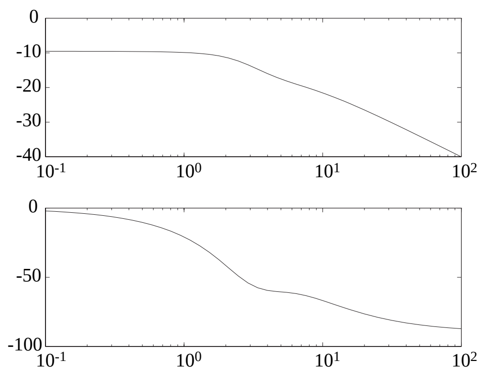
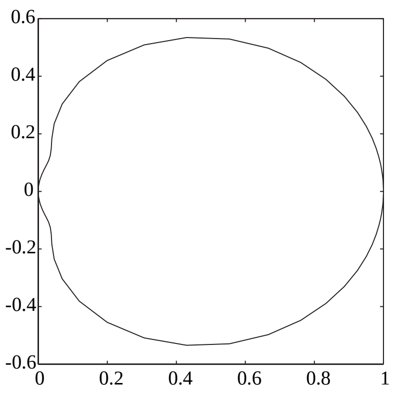
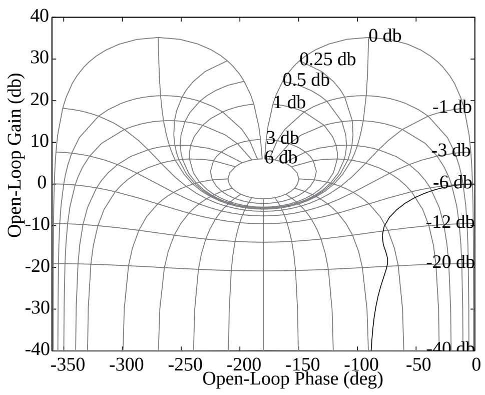


Figure A.23. *The modified graph.*

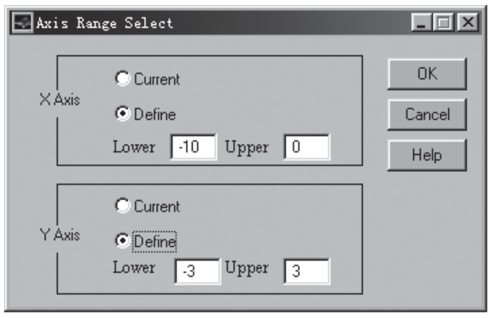


(a) Nyquist plots



(b) Nichols chart

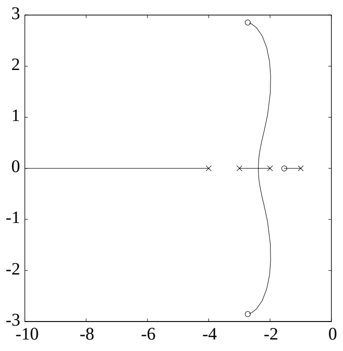
Figure A.24. *Frequency responses.*


Axis Range Select

X Axis
Current
[✓ Define]
Lower -10 Upper 0

Y Axis
Current
[✓ Define]
Lower -3 Upper 3

(a) zoom dialog box



(b) root locus

Figure A.25. *Root locus analysis.*

A.4.2 Time Domain Analysis

The step and impulse responses of the system can be obtained directly from the menu Analysis | Step response, and Analysis | Impulse response, respectively. For instance, the

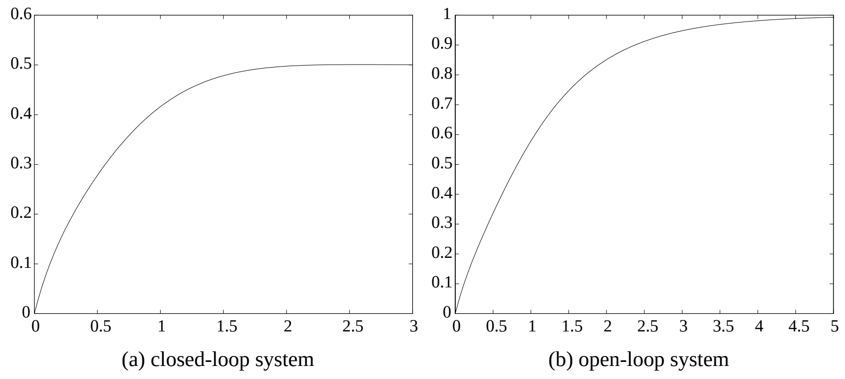
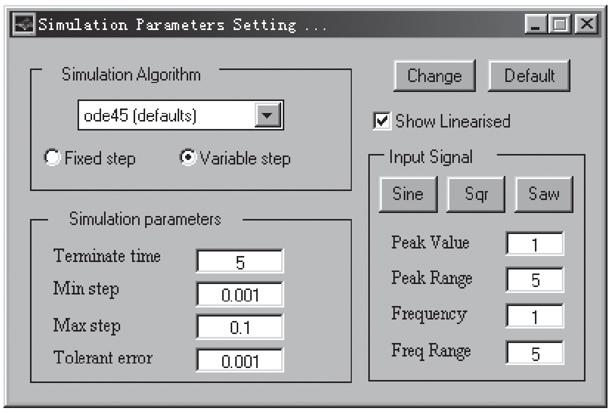


Figure A.26. *Step response analysis.*



Simulation Parameters Setting ...

Simulation Algorithm: ode45 (defaults)

Fixed step [✓ Variable step]

Simulation parameters

Terminate time	5
Min step	0.001
Max step	0.1
Tolerant error	0.001

Change Default

[✓] Show Linearised

Input Signal

Sine Sqr Saw

Peak Value	1
Peak Range	5
Frequency	1
Freq Range	5

Figure A.27. *Simulation parameter setting dialog box.*

step response of the system can be obtained as shown in Figure A.26(a). This step response shown in Figure A.26(a) is the closed-loop step response. One can obtain the open-loop step response of the system by selecting the relevant submenu item in the Analysis menu and the open-loop step response of the system can then be redrawn in the step response window as shown in Figure A.26(b).

For nonlinear systems, one can also specify the type of input signals, via the Options | Simulation parameters menu item in the relevant graphics window. A dialog box will appear as shown in Figure A.27 which prompts the user to specify the input signals as well as the simulation parameters. For instance, when studying the system with the Simulink model, to display the step response of the linearized system and that of the original system,

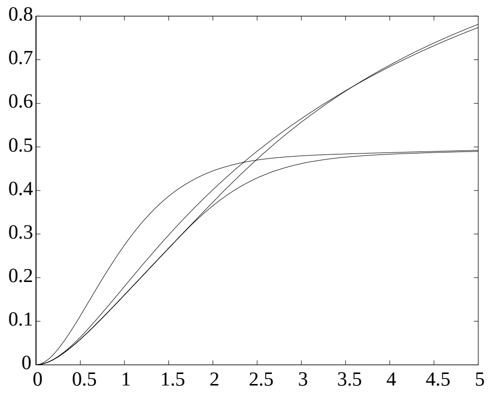
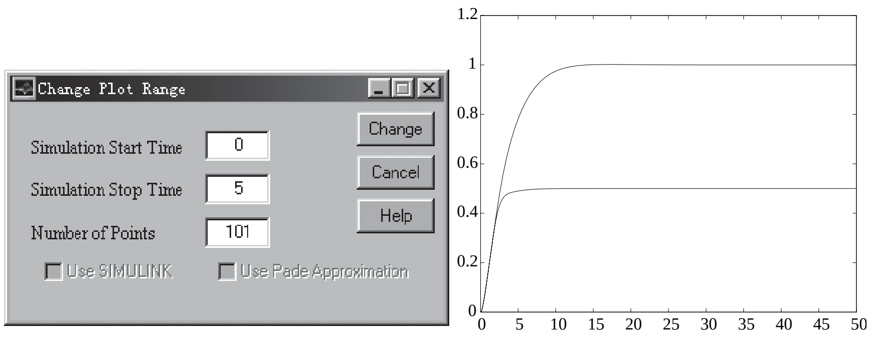


Figure A.28. *Step responses of a nonlinear system with linearization.*


Change Plot Range

Simulation Start Time	0
Simulation Stop Time	5
Number of Points	101

☐ Use SIMULINK ☐ Use Pade Approximation

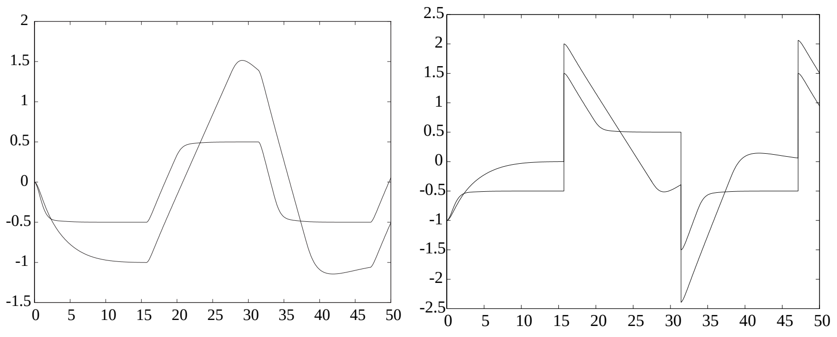
(a) plot range setting (b) with a new time range

Figure A.29. *Time range modifications.*

check the Show Linearised box. The time response of the system can then be displayed as shown in Figure A.28.

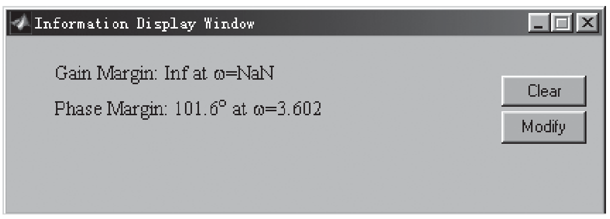
The plot range can also be set by the Options | Plot range menu item in the graphics window. A dialog box, shown in Figure A.29(a), prompts the user to select a new plot range. For instance, the user can set a new terminating time at 50, and the new system responses are then obtained as shown in Figure A.29(b).

Other signal types apart from the step and impulse signals can also be applied. For instance, the user can select square wave, saw tooth, wave and sine wave by using the dialog box shown in Figure A.27. Other parameters such as the frequency of the signal can also be changed. The time response to a square wave input is shown in Figure A.30(a). To display other signals such as the error signal $e(t)$, select the Options | Other signals menu item in the graphics window and click the error signal $e(t)$ in the block diagram of the feedback system. The error signal for a step input can then be obtained as shown in Figure A.30(b).



(a) square wave input response (b) error signal

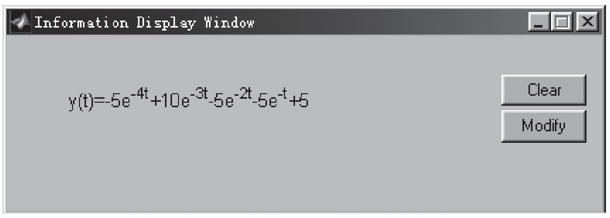
Figure A.30. *Time response of other signals.*



Gain Margin: Inf at ω=NaN

Phase Margin: 101.6° at ω=3.602

Figure A.31. *Gain and phase margins.*



$y(t)=-5e^{-4t}+10e^{-3t}-5e^{-2t}-5e^{-t}+5$

Figure A.32. *Analytical closed-loop step response.*

A.4.3 System Properties Analysis

The stability property, gain and phase margins, and the analytical solutions to step and impulse signals can also be obtained through the menu system. For instance, for the nonlinear system model, the gain and phase margins to the linearized model can be obtained as shown in Figure A.31, and the analytical solutions to the step response of the system can then be shown as in Figure A.32.

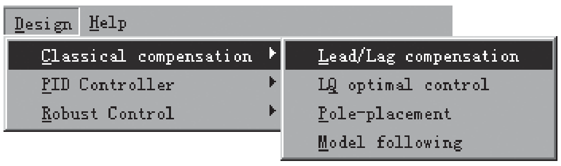
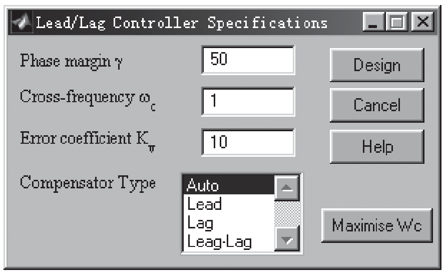


Figure A.33. *System design menu.*


Lead/Lag Controller Specifications

Phase margin γ: 50

Cross-frequency ω_c: 1

Error coefficient K_v: 10

Compensator Type: Auto (Auto, Lead, Lag, Leag-Lag)

Figure A.34. *Lead-Lag compensator dialog box.*

A.5 Controller Design Examples

A.5.1 Model-Based Controller Designs

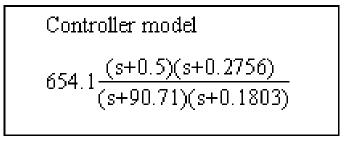
We shall use the phase lead-lag controller design problem as an example to illustrate the controller design for a given plant model via CtrlLAB. The model-based controller design menu is shown in Figure A.33, and it can be seen that several model-based design algorithms can be selected within the menu, as discussed in Chapter 5. For instance, with a typical lead-lag controller design dialog box, shown in Figure A.34, the user is requested to enter the parameters such as the expected phase margin γ, the crossover frequency ω_c, and the steady-state error tolerance K_v.

Let us try a plant model given by $G(s) = 1/[s(s+1)(0.2s+1)]$. Set the expected phase margin $\gamma = 50°$, the crossover frequency $\omega_c = 5$ rad/sec, and the steady-state error tolerance $K_v = 100$. Then, a lead-lag compensator can be designed as shown in Figure A.35(a). With a proper menu selection, the controller can be shown in the factorized form as in Figure A.35(a). The Bode diagrams of the system before and after lead-lag compensation can be obtained using the Analysis | Bode Diagram menu item, as shown in Figure A.35(b).

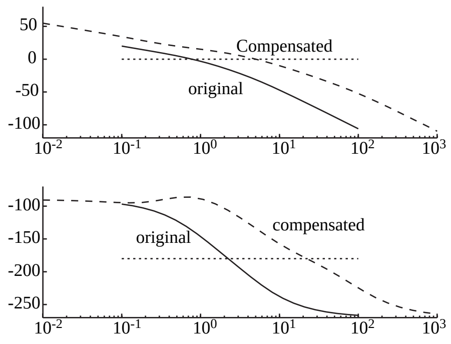
Via CtrlLAB, it is also very easy to design the LQ optimal controller and the pole-placement controller with either full state feedback or observer-based structures. The straightforward model-based controllers can also be designed with CtrlLAB.

A.5.2 Design of PID Controllers

Consider the PID controller design problem with the plant model $G(s) = 10/[(s+1)(s+2)(s+3)(s+4)]$ entered via CtrlLAB. By the Design | PID Controller menu item, the design menu will appear as shown in Figure A.36. It can be seen that different PID controller design algorithms have been implemented within CtrlLAB. The “one-shot” submenu item


Controller model

$$654.1\frac{(s+0.5)(s+0.2756)}{(s+90.71)(s+0.1803)}$$



(a) lead-lag controller (b) Bode diagram comparison

Figure A.35. *A lead-lag compensator.*

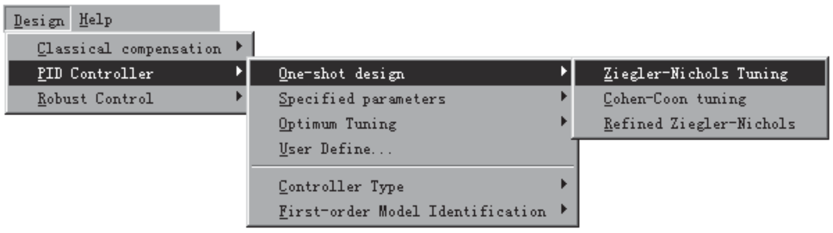
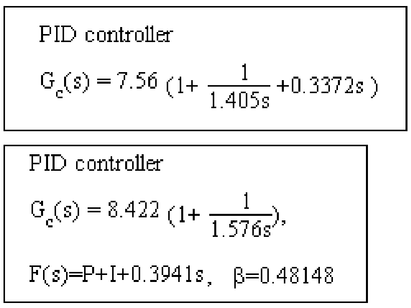


Figure A.36. *Main menu for PID controller design.*

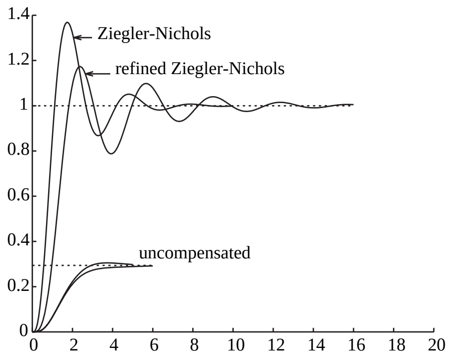

PID controller

$$G_c(s) = 7.56\ (1+\frac{1}{1.405s}+0.3372s\ )$$

PID controller

$$G_c(s) = 8.422\ (1+\frac{1}{1.576s}),$$

$F(s)$=P+I+0.3941s, β=0.48148



(a) controller parameters (b) step response comparison

Figure A.37. *Ziegler–Nichols PID controller.*

in Figure A.36 means that the PID controller can be designed directly from the known plant model with no other extra specification needed. One may design a PID controller using the Ziegler–Nichols algorithm by selecting Design | PID controller | One-shot design | Ziegler–Nichols Tuning. This will immediately generate the PID controller as shown in Figure A.37(a). Furthermore, the refined Ziegler–Nichols controller can be designed, as also shown in Figure A.37(a), when the One-shot design | Refined Ziegler–Nichols menu is selected. By the Analysis | Step response menu item, the closed-loop step response of

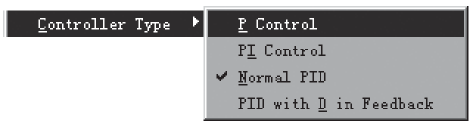


Figure A.38. *PID controller structures.*

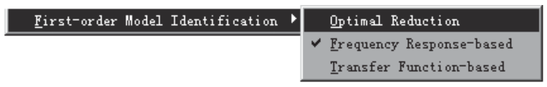


Figure A.39. *FOPDT model fitting methods.*

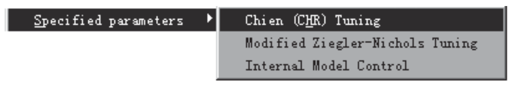


Figure A.40. *PID with specified parameters.*

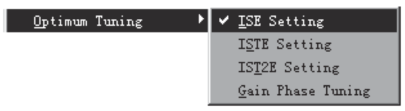


Figure A.41. *Optimum PID controller design.*

the system will be obtained as in Figure A.37(b) where it is shown together with the step response of the uncompensated system.

Apart from the standard PID controllers, other similar structures such as the P controller, the PI controller, and the PID controller with D in the feedback loop, can also be designed, which can be selected from the Design PID Controller | Controller Type menu item as shown in Figure A.38. We know that the PID controller parameter setting is based on the first-order plus dead time (FOPDT) model. Given a high-order plant model, we can select different approaches to fit the original plant model by a standard first-order model with dead time. The fitting algorithms can be selected from the menu shown in Figure A.39.

PID controllers can also be designed with other algorithms using the Specified parameters and Optimum Tuning menu items as shown in Figures A.40 and A.41, respectively.

With the above different tuning algorithms, we can design PID controllers that have better performance. For instance, the suboptimal first-order approximation to the plant model can be obtained using menu item First-order model identification | Optimal reduction, and from this an optimum PID controller can be designed. Using these controllers,

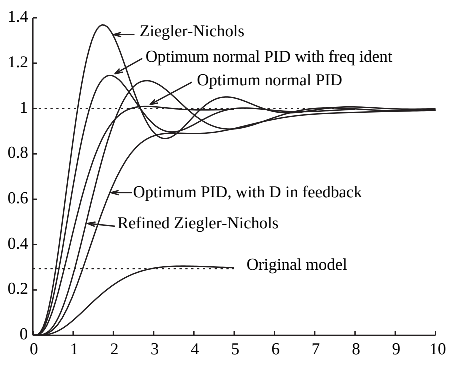


Figure A.42. *Step response comparison of different PID controllers.*

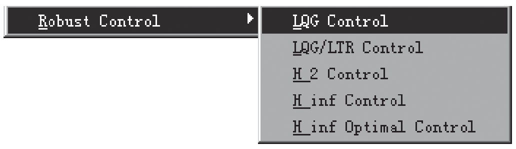


Figure A.43. *Robust control design menu.*

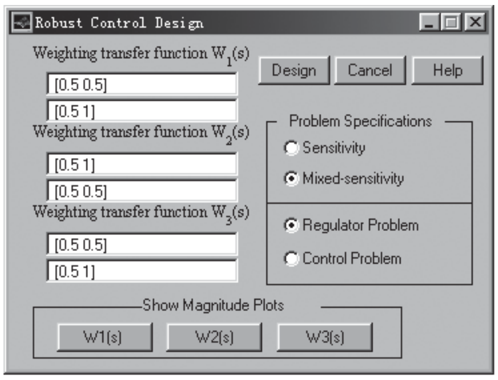
the closed-loop step responses are then compared as in Figure A.42. It can be seen that performance can be significantly improved, compared to the results from other "one-shot" PID controllers.

A.5.3 Robust Controller Design

In this section, only the $\mathcal{H}_\infty$ controller design via CtrlLAB will be demonstrated, although other design problems can also be solved in CtrlLAB. The example we shall use is the double integrator plant model as given in Example 7.16. The design submenus for the robust controllers can be obtained by selecting the Design | Robust Control menu item as shown in Figure A.43.

To get an $\mathcal{H}_\infty$ optimal controller, select the Design | Robust Control | H_inf Optimal Control menu item to obtain the dialog box shown in Figure A.44. Specify various weighting functions $W_1(s)$, $W_2(s)$, and $W_3(s)$ in the dialog box. To design an $\mathcal{H}_\infty$ controller for the sensitivity problem, check Sensitivity so that a new dialog box will appear as shown in Figure A.45(a). In Figure A.45(a), the expected order and the natural frequency for the ITAE standard reference model should be entered. For instance, if one selects $n = 2$ and $\omega_n = 10$ rad/sec, an optimal $\mathcal{H}_\infty$ controller can be designed as shown in Figure A.45(b).

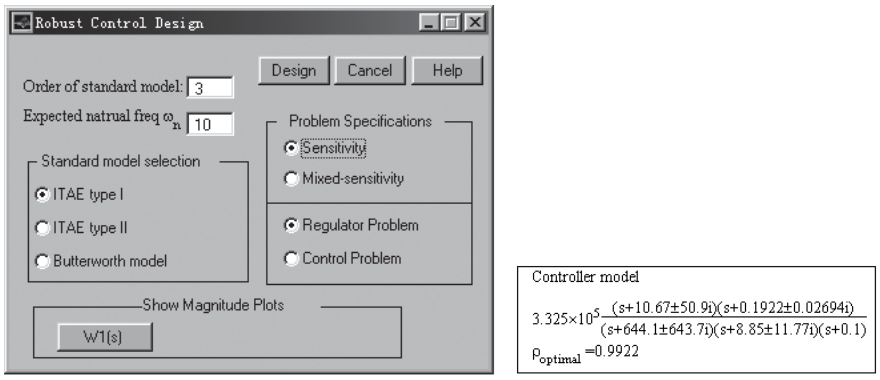
The Nichols charts and the closed-loop step response of the system can then be obtained as shown in Figures A.46(a) and (b), respectively. Other types of robust controllers, such as the $\mathcal{H}_2$ controller and the LQG/LTR controllers, can also be designed and analyzed with little effort using the menus and dialog boxes.



Weighting transfer function $W_1(s)$: [0.5 0.5] [0.5 1]
Weighting transfer function $W_2(s)$: [0.5 1] [0.5 0.5]
Weighting transfer function $W_3(s)$: [0.5 0.5] [0.5 1]

Problem Specifications: Sensitivity [✓ Mixed-sensitivity]
[✓ Regulator Problem] Control Problem

Figure A.44. *$\mathcal{H}$-norm-based dialog box.*



Order of standard model: 3
Expected natrual freq ω_n: 10

Standard model selection: [✓ ITAE type I] ITAE type II Butterworth model

Problem Specifications: [✓ Sensitivity] Mixed-sensitivity
[✓ Regulator Problem] Control Problem

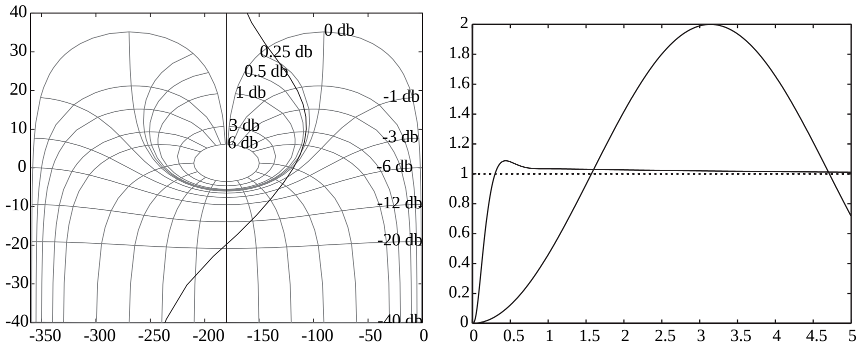
Controller model

$$3.325\times10^5\frac{(s+10.67\pm50.9i)(s+0.1922\pm0.02694i)}{(s+644.1\pm643.7i)(s+8.85\pm11.77i)(s+0.1)}$$

$\rho_{\text{optimal}}=0.9922$

(a) dialog box for the sensitivity problem (b) optimal $\mathcal{H}_\infty$ controller

Figure A.45. *Robust control design results.*



(a) Nichols charts (b) closed-loop step response

Figure A.46. *Robust control system analysis.*

A.6 Graphical Interface-Based Tools

Two useful graphics-based tools are provided in CtrlLAB which can be used to process matrices and figures, respectively. In the following subsections, detailed descriptions of these two programs will be given.

A.6.1 A Matrix Processor

A matrix processor, `MatxProc()` is developed which can be used to process and edit matrices and state space models, and perform various kinds of matrix analyses in a visual way. The GUI facilities are extensively used to make the matrix processor very flexible and easy to use.

When `MatxProc` is typed in the MATLAB prompt, a GUI will appear as shown in Figure A.47. The program can also be called from within CtrlLAB. In MATLAB, `MatxProc()` can be called using the format `MatxProc(A)`, where `A` is a given matrix, or simply using `MatxProc`.

The File | New matrix menu can be selected to create a new matrix. The dialog box shown in Figure A.48 will appear to prompt the user to select from different matrix templates. For instance, if one selects a Hilbert matrix with 3 rows, the matrix will then be created by `MatxProc` as shown in Figure A.49.

Various display formats are allowed in `MatxProc()`. The user can select the Format menu as shown in Figure A.50(a). It can be seen that the user can specify different display precisions (high, normal, or rational), different alignment requirements (left, right, or center), and different truncating thresholds. For instance, the high precision display is given in Figure A.50(b), with part of the matrix elements hidden due to the limited size of the window. The hidden part of the matrix can be displayed via the horizontal scroll bar. The matrix can also be displayed in rational number format.

A matrix displayed can be analyzed and processed within `MatxProc()`. For instance, to analyze the matrix, simply select the Analysis to obtain the menu appearing in Figure A.51. To get the parameters of the given matrix, select the Analysis | Matrix

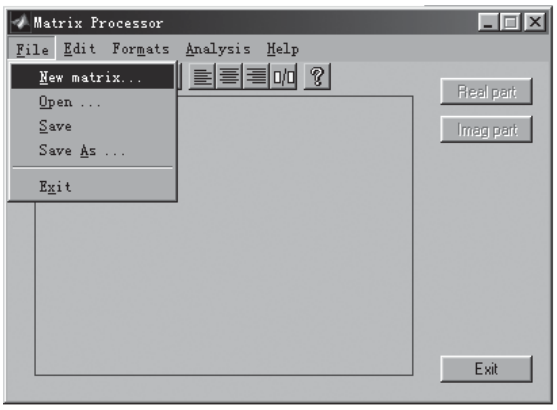


Figure A.47. *A matrix processor interface.*

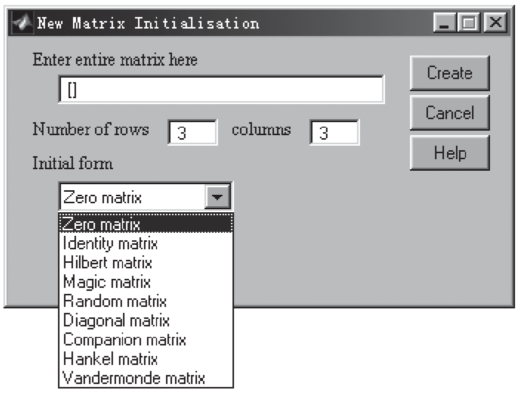
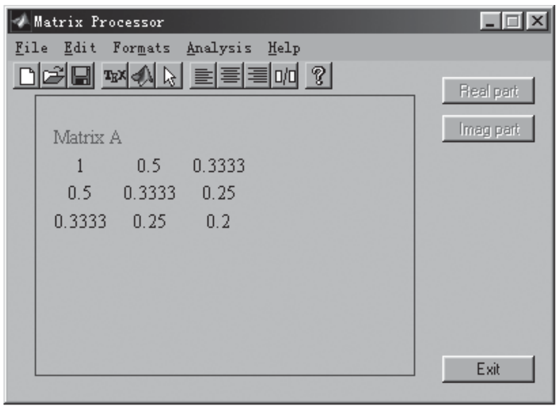
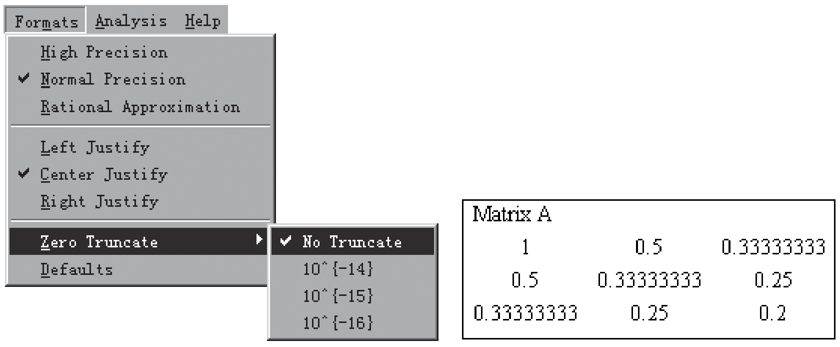


Figure A.48. *Matrix creating dialog box.*



```
Matrix A
  1        0.5     0.3333
  0.5     0.3333   0.25
0.3333    0.25     0.2
```

Figure A.49. *Creating a new matrix.*



```
Matrix A
     1            0.5        0.33333333
    0.5       0.33333333       0.25
0.33333333       0.25           0.2
```

(a) format menu (b) high precision display

Figure A.50. *Display formats of a matrix.*

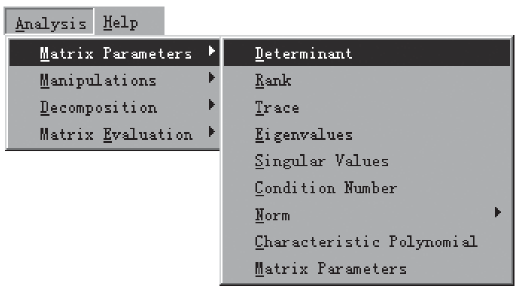
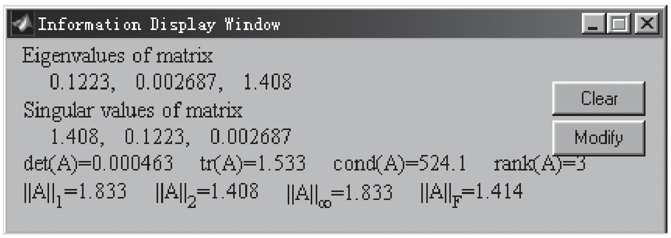


Figure A.51. *Matrix analysis menu.*

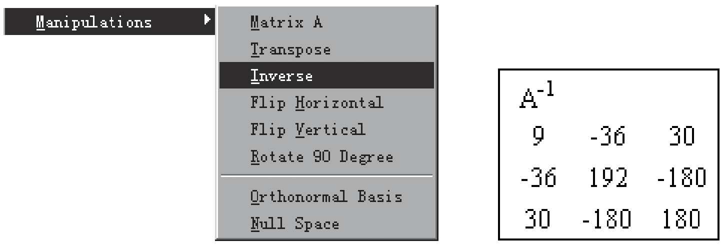


```
Eigenvalues of matrix
   0.1223,  0.002687,  1.408
Singular values of matrix
   1.408,  0.1223,  0.002687
```

det(A)=0.000463 tr(A)=1.533 cond(A)=524.1 rank(A)=3

$||A||_1$=1.833 $||A||_2$=1.408 $||A||_\infty$=1.833 $||A||_F$=1.414

Figure A.52. *Matrix parameters display.*



A^{-1}

9	-36	30
-36	192	-180
30	-180	180

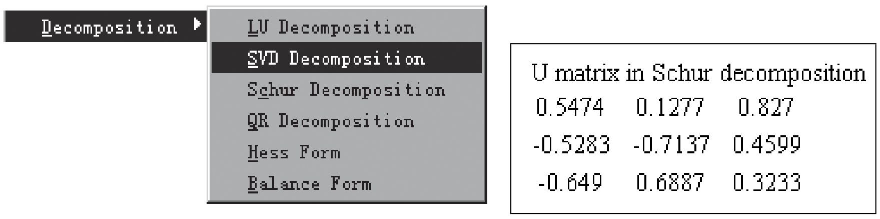
(a) manipulation menu (b) inverse matrix

Figure A.53. *Matrix manipulations.*

Parameters menu item. The analysis results will be obtained and displayed in the Information Display Window as shown in Figure A.52. Other analysis tasks such as evaluating the determinant, trace, norm, characteristic polynomial of the matrix can also be performed using the Analysis menu.

Matrix manipulation such as matrix inversion and rotation can be performed within `MatxProc()`. To manipulate the matrix, select the Analysis | Manipulations menu as shown in Figure A.53(a) to easily obtain, for example, the inversion of the matrix shown in Figure A.53(b).

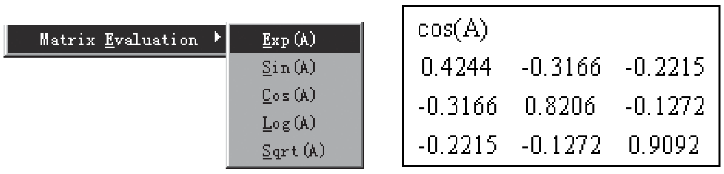
Different decompositions for a given matrix can also be obtained, such as the QR decomposition, LU decomposition, singular value decomposition (SVD), etc. The Analysis | Decomposition menu is shown in Figure A.54(a), where the $\boldsymbol{U}$ matrix of the Schur decomposition can easily be obtained by selecting the relevant menu item, and the results are shown in Figure A.54(b). In addition, the button labeled T matrix in the GUI prompts the user to display the other matrix, for example, the $\boldsymbol{T}$ matrix, such that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{T}\boldsymbol{U}^{\mathrm{T}}$.



```
U matrix in Schur decomposition
 0.5474    0.1277    0.827
-0.5283   -0.7137    0.4599
-0.649     0.6887    0.3233
```

(a) decomposition menu (b) $\boldsymbol{U}$ matrix

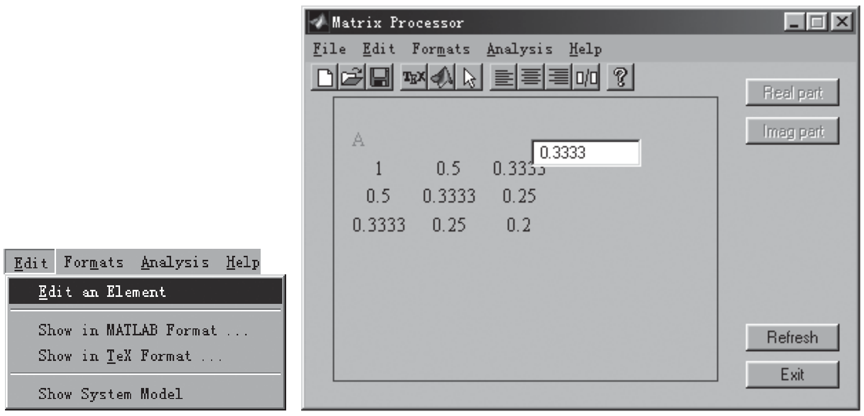
Figure A.54. *Matrix decompositions.*



```
cos(A)
 0.4244   -0.3166   -0.2215
-0.3166    0.8206   -0.1272
-0.2215   -0.1272    0.9092
```

(a) matrix evaluation menu (b) cosine function

Figure A.55. *Matrix function evaluations.*



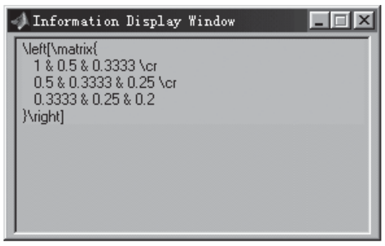
```
A                 0.3333
  1       0.5     0.3333
  0.5     0.3333  0.25
  0.3333  0.25    0.2
```

(a) matrix edit menu (b) matrix editing interface

Figure A.56. *Matrix editing facilities.*

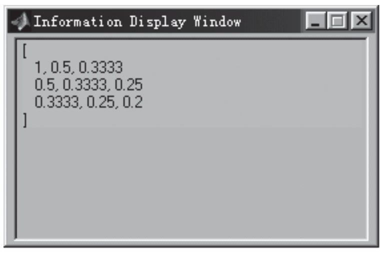
Matrix function evaluations can be performed within `MatxProc()` by selecting the Analysis | Matrix Evaluation menu. Contents of the menu are displayed in Figure A.55(a). When the user selects the Cos(A) function display, the cosine of matrix $\boldsymbol{A}$ can be obtained as shown in Figure A.55(b).

A matrix can be edited using the Edit menu as shown in Figure A.56(a). By the Edit | Edit an Element menu item, the cursor will be changed to the cross sign, which prompts the user to select a matrix element. Once the user has selected an element to edit, the value of the element will be entered into the edit box for modification, as shown in Figure A.56(b). Once the edit process is done, the user can press the Accept button to confirm the change.



```
\left[\matrix{
  1 & 0.5 & 0.3333 \cr
  0.5 & 0.3333 & 0.25 \cr
  0.3333 & 0.25 & 0.2
}\right]
```

(a) TEX format



```
[
  1, 0.5, 0.3333
  0.5, 0.3333, 0.25
  0.3333, 0.25, 0.2
]
```

(b) MATLAB format

Figure A.57. *Matrix display in other formats.*

The matrix can be shown in other formats as well, such as the TEX format and the MATLAB format. This is particularly useful in dealing with large and complicated matrices. For instance, the TEX format of the matrix can be obtained by selecting the Edit | Show in TeX Format menu item, and the result is as shown in Figure A.57(a), while the MATLAB format of the matrix is shown in Figure A.57(b).

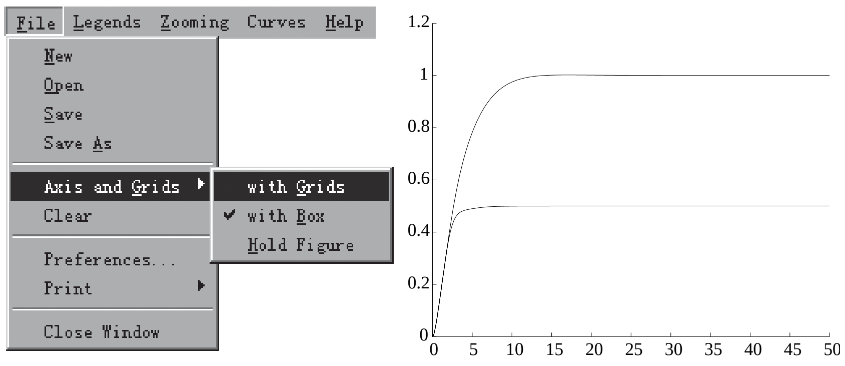
A.6.2 A Graphical Curve Processor

The graphical curve processor is not currently an independent MATLAB function. It has been integrated into CtrlLAB. It is mainly used to “decorate” the graphs obtained using CtrlLAB to any degree of complexity. It can be used to do simple things such as add or remove grids, add arrows, add floating legends to the graph, etc. Most of the figures in this book used this unique graphical curve processor within CtrlLAB. We remark that, although the current version of MATLAB has provided a plot editing toolbar for various graph editing utilities, the graphical curve processor within CtrlLAB has been working similarly and more powerfully with earlier versions of MATLAB (since version 4.2c) and is compatible with versions 5.x and 6.x. The ultimate objective of CtrlLab is to minimize user effort.

An Option menu in the standard MATLAB graphics window allows for some of the useful facilities to be called; this menu is shown in Figure A.58(a). For instance, via the Options | Axis and Grid | with Boxes off and Options | Axis and Grid | with Grid off menu items, the time response graph will then be changed to the display format shown in Figure A.58(b), where the grids and boxes are turned off.

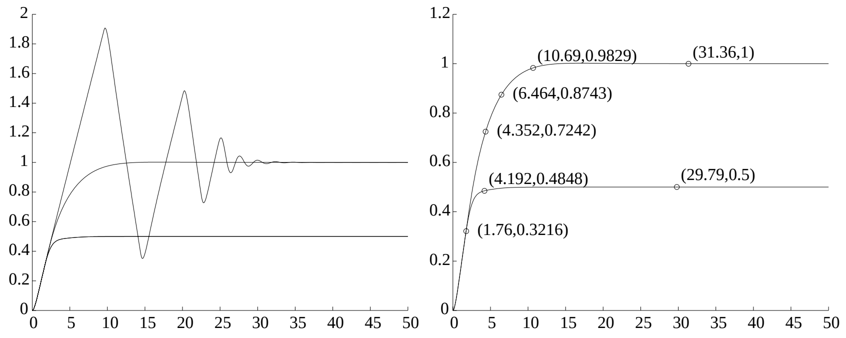
Note that, to turn off the grids, we can type `grid off` within the MATLAB command line. However, our objective here is to avoid such a user involvement. At this point, we remark again that CtrlLAB is designed for linear feedback control system analysis and design by *only mouse clicks* and some essential numeric key strokes. Great efforts have been made to minimize the user involvement in the analysis and design of feedback control systems. The Matrix Processor and Graph Processor described in this section are also part of the efforts to achieve this goal.

To draw several curves together with a common coordinate, select the Options | Axis and Grid | Hold on menu item to hold the current graph coordinate and then display another curve on the current plot. This is demonstrated in Figure A.59(a).



(a) Options menu (b) curve without box and grid

Figure A.58. *Graphics processor menu and results.*



(a) graph holding (b) cursor positioning

Figure A.59. *Screen hold and cursor.*

To cancel the hold protection, select the Options | Axis and Grid | Hold off menu item. To locate the specific points on the graph, use the Options | Cursor positions menu item. For instance, the curves with some points selected and marked are shown in Figure A.59(b).

Furthermore, various legends can be added to the graphs. The Options | Legends menu is shown in Figure A.60, where one can select to add, move, or edit text strings on the graphs, and also to draw lines or lines with arrows on the graph.

Two text legends are added on the graph shown in Figure A.61(a), and several lines and arrows can be further added on the graph as shown in Figure A.61(b). It can be seen that the legends (including lines and arrows) can be added or edited freely using the facilities provided. The user can also remove the legends by selecting Options | Legends | Delete a Legend to remove an existing legend.

The properties of the legends can be modified if the user selects the Legends | Properties menu item, and a dialog box for assigning legend properties will be displayed as shown in Figure A.62(a). With proper settings, the modified version of the graph with different fonts, and line types will be obtained as shown in Figure A.62(b).

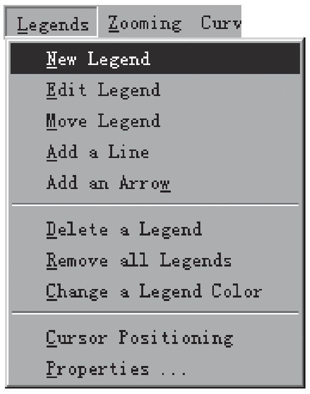
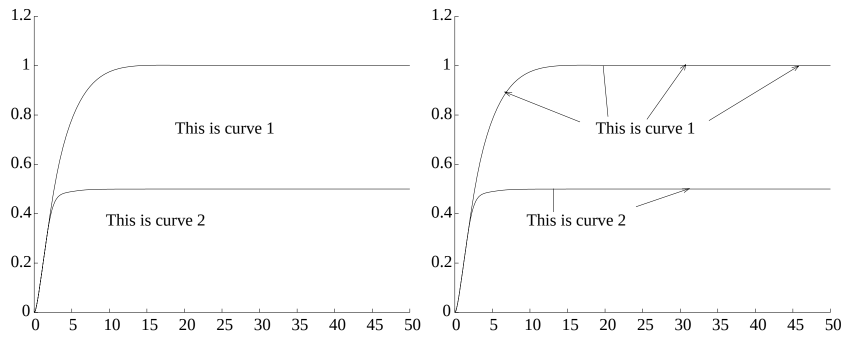
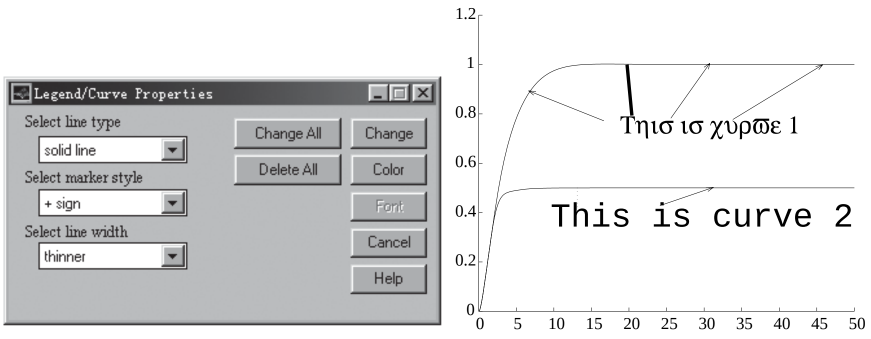


Figure A.60. *Legends menu.*



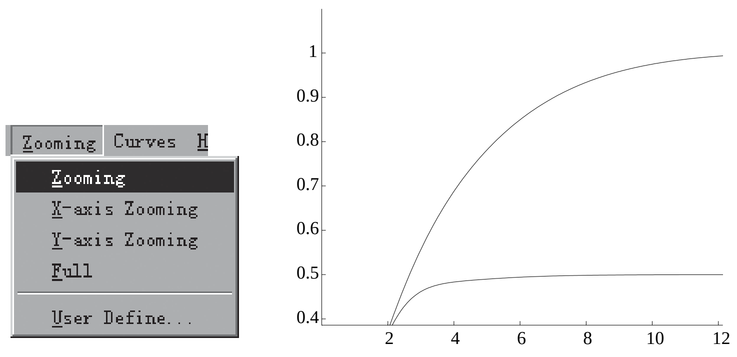
(a) examples of legends (b) examples of arrows and lines

Figure A.61. *Adding more legends on graphs.*



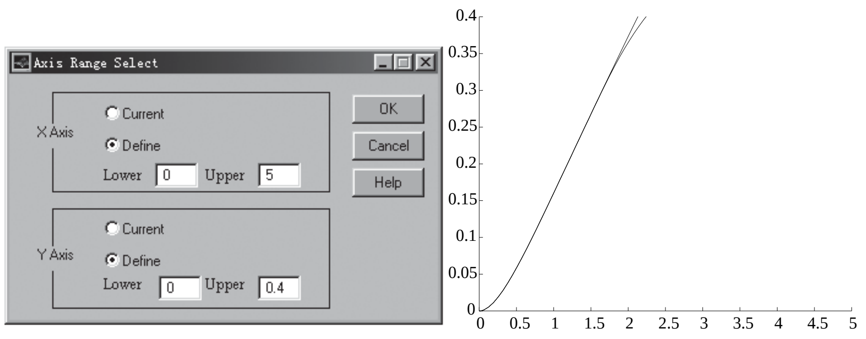
(a) legend properties dialog box (b) modified legends

Figure A.62. *Changing the properties of legends.*



(a) zoom menu (b) zoomed graphic display

Figure A.63. *Zoom facilities.*


Axis Range Select

X Axis: Current [✓ Define] Lower 0 Upper 5

Y Axis: Current [✓ Define] Lower 0 Upper 0.4

(a) axis specification dialog box (b) zoomed graphic display

Figure A.64. *Axis range specifications.*

The user may also change the view in the graph window by selecting the Options | Zooming menu item as shown in Figure A.63(a), which allows the user to change the current coordinates using a mouse. For instance, the user can redefine the range for display by dragging the mouse, and the results can then be displayed as shown in Figure A.63(b).

Moreover, using the Zooming | User Define menu item, the dialog box shown in Figure A.64(a) will pop up to allow the user to select a reasonable display range. If the plot range in Figure A.64(a) is used, the zoomed output will be displayed as shown in Figure A.64(b).

Problems

1. Use the following plant models to test the previously described analysis and design tasks using CtrlLAB:

   (a) $G(s) = \dfrac{50000}{(s+1)(s+2)(s+3)(s+4)(s+5)(s+6)(s+7)(s+8)}$.

(b) $$\dot{\boldsymbol{x}} = \begin{bmatrix} 2.25 & -5 & -1.25 & -0.5 \\ 2.25 & -4.25 & -1.25 & -0.25 \\ 0.25 & -0.5 & -1.25 & -1 \\ 1.25 & -1.75 & -0.25 & -0.75 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 4 \\ 2 \\ 2 \\ 0 \end{bmatrix}u, \quad y = x_1 + 5x_2.$$

(c) The DC drive system given in Example 2.11. Use both the direct method and the Simulink method to create the system model.

2. Analyze the system matrix in problem 1(b). Find the norms, determinant, eigenvalues, and characteristic polynomial of $\boldsymbol{A}$, and do LU, QR, SVD decomposition of $\boldsymbol{A}$ within CtrlLAB. Find the matrices $e^{\boldsymbol{A}}$, $\sin(\boldsymbol{A})$, and $l og(\boldsymbol{A})$.

3. Try to reproduce Figure 3.14(a) by using the graphics processor.

Bibliography

[1] Callier F. M., Desoer C. A. *Multivariable Feedback Systems*. New York: Springer-Verlag, 1982

[2] Freudenberg J. S., Looze D. P. *Frequency Domain Properties of Scalar and Multivariable Feedback Systems*, Lecture Notes in Control and Information Sciences, volume 104. Berlin: Springer-Verlag, 1988

[3] Maciejowski J. M. *Multivariable Feedback Design*. Wokingham, England: Addison-Wesley, 1989

[4] Postlethwaite I., MacFarlane A. G. J. *A Complex Variable Approach to the Analysis of Linear Multivariable Feedback Systems*. Berlin: Springer-Verlag, 1979

[5] Skogestad S., Postlethwaite I. *Multivariable Feedback Control*: *Analysis and Design*. Chichester, England: John Wiley & Sons, 1996

[6] Vardulakis A. I. G. *Linear Multivariable Control — Algebraic Analysis and Synthesis Methods*. Chichester, England: John Wiley & Sons, 1991

[7] Wonham W. M. *Linear Multivariable Control — A Geometric Approach*, Lecture Notes in Economics and Mathematical Systems, volume 101. Berlin: Springer-Verlag, 1974

[8] Mayr O. *The Origins of Feedback Control*. Cambridge, MA: MIT Press, 1970

[9] Minorsky N. *Directional stability of automatically steered bodies*. Journal of the American Society of Naval Engineering, 1922, 34(2):280–309

[10] Ziegler J. G., Nichols N. B. *Optimum settings for automatic controllers*. Transactions of the ASME, 1942, 64:759–768

[11] Nyquist H. *Regeneration theory*. Bell System Technology Journal, 1932, 11:126–147

[12] Bode H. W. *Network Analysis and Feedback Amplifier Design*. Princeton, NJ: Van Nostrand, 1945

[13] James H. M., Nichols N. B., Phillips R. S. *Theory of Servomechanisms*, MIT Radiation Laboratory Series, volume 25. New York: McGraw–Hill, 1947

[14] Evans W. R. *Graphical analysis of control systems*. Transactions of the AIEE, 1948, 67:547– 551

[15] Pontryagin L. S., Boltyanskii V. G., Gamkrelidze R. V., Mischenko E. F. The Mathematical Theory of Optimal Processes. New York: Interscience Publishers, 1962. Translated from the Russian by K. N. Trirogoff

[16] Bellman R. *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957

[17] Kalman R. E. *On the general theory of control systems*. IRE Transactions on Automatic Control, 1959, 4(3):110. Abstract. Full paper published in Proceedings of the 1st IFAC Congress, Moscow, 1960

[18] Kalman R. E. *Mathematical description of linear dynamical systems*. SIAM Journal of Control, 1963, 1(2):152–192

[19] Kalman R. E. *When is a linear control system optimal?* Transactions of ASME Journal of Basic Engineering Series D, 1964, 86:51–60

[20] Doyle J. C., Stein G. *Robustness with observers*. IEEE Transactions on Automatic Control, 1979, AC-24:607–611

[21] Zhang Z., Freudenberg J. S. *Loop transfer recovery for nonminimum phase plants*. IEEE Transactions on Automatic Control, 1990, 35(5):547–553

[22] Zames G. *Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms and approximate inverses*. IEEE Transactions on Automatic Control, 1981, AC-26(4):301–320

[23] Doyle J. C., Glover K., Khargoneckar P. P., Francis B. A. *State space solutions to standard H_2 and H_∞ control problems.* IEEE Transactions on Automatic Control, 1989, 34(8):831–847

[24] Melsa J. L., Jones S. K. *Computer Programmes for Computational Assistance in the Study of Linear Control Theory*. New York: McGraw–Hill, 1973

[25] Moler C. B. *MATLAB — An Interactive Matrix Laboratory*. Technical Report 369, University of New Mexico, Albuquerque, NM, 1980

[26] Åström K. J. *Computer aided tools for control system design*. In Jamshidi M, Herget C, eds., *Computer-Aided Control System Engineering*. Amsterdam: Elsevier Science Publishers B. V, 1985, 3–40

[27] *Using MATLAB version* 6.1. The MathWorks, Natick, MA, 2001

[28] Xue D. *Computer-aided Design of Control Systems with MATLAB.* Beijing: Tsinghua University Press (in Chinese), 1996

[29] Xue D., Chen Y. Q. *MATLAB/Simulink Based System Simulation Techniques.* Beijing: Tsinghua University Press (in Chinese), 2002

[30] Moore B. *Principal component analysis in linear systems: controllability, observability, and model reduction*. IEEE Transactions on Automatic Control, 1981, 26:17–32

[31] Ljung L. *System Identification — Theory for the User*. 2nd edition. Upper Saddle River, NJ: PTR Prentice Hall, 1999

[32] Akaike H. *A new look at the statistical model identification*. IEEE Transactions on Automatic Control, 1974, 19(6):716–723

[33] Levy E. C. *Complex-curve fitting*. IRE Transactions on Automatic Control, 1959, 4:37–43

[34] Andresen T. *A logarithmic-amplitude polar diagram*. Modeling, Identification and Control, 2001, 22(2):65–72

[35] Davison E. J. *A method for simplifying linear dynamic systems*. IEEE Transactions on Automatic Control, 1966, 11:93–101

[36] Atherton D. P., Borne P. *Concise Encyclopedia of Modelling and Simulation*. New York: Pergamon Press, 1992

[37] Bultheel A., van Barel M. *Padé techniques for model reduction in linear system theory: A survey*. Journal of Computational and Applied Mathematics, 1986, 14:401–438

[38] Decoster M., van Cauwenberghe A. R. *A comparative study of different reduction methods (Parts 1 & 2)*. Journal A, 1976, 17:68–74;125–134

[39] Hutton M. F. *Routh approximation for high-order linear systems*. In Proceedings of the 9th Allerton Conference. 1971, 160–169

[40] Shamash Y. *Linear system reduction using Padé approximation to allow retention of dominant modes*. International Journal of Control, 1975, 21:257–272

[41] Lucas T. N. *Some further observations on the differential method of model reduction*. IEEE Transactions on Automatic Control, 1992, 37:1389–1391

[42] Chen C. F., Chang C. Y., Han K. W. *Model reduction using the stability-equation method and the continued fraction method*. International Journal of Control, 1980, 32:81–94

[43] Hu X. H. *FF-Padé method of model reduction in frequency domain*. IEEE Transactions on Automatic Control, 1987, 32:243–246

[44] Hwang C., Lee Y. *Multi-frequency Padé approximation via Jordan continued-fraction expansion*. IEEE Transactions on Automatic Control, 1989, 34:444–446

[45] Xue D., Atherton D. P. *An optimal model reduction algorithm for linear systems*. In Proceedings of the American Control Conference. Boston, MA, 1991, 2128–2129

[46] Xue D. *Model Reduction Techniques and Applications*. Shenyang, China: Lecture Notes of Northeastern University, 1996

[47] Xue D., Atherton D. P. *A suboptimal reduction algorithm for linear systems with a time delay*. International Journal of Control, 1994, 60(2):181–196

[48] Gruca A., Bertrand P. *Approximation of high-order systems by low-order models with delays*. International Journal of Control, 1978, 28:953–965

[49] Glover K. *All optimal Hankel-norm approximations of linear multivariable systems and their $\mathcal{L}^{\infty}$-error bounds*. International Journal of Control, 1984, 39:1115–1193

[50] Stahl H., Hippe P. *Comments on "FF-Padé method of model reduction in frequency domain."* IEEE Transactions on Automatic Control, 1988, 33:415–416

[51] Atherton D. P. *Nonlinear Control Engineering — Describing Function Analysis and Design*. London: Van Nostrand Reinhold, 1975

[52] *Using Simulink Version* 4.1. The MathWorks, Natick, MA, 2001

[53] Franklin G. F., Powell J. D., Workman W. *Digital Control of Dynamic Systems*. 3rd edition. Reading, MA: Addison Wesley, 1988

[54] Frederick D. K., Rimer M. *Benchmark problem for CACSD packages*. In Abstracts of the Second IEEE Symposium on Computer-Aided Control System Design. Santa Barbara, CA, 1985

[55] Dorato P. *Linear Quadratic Control — An Introduction*. New York: McGraw–Hill, 1995

[56] Balasubramanian R. *Continuous Time Controller Design*. Stevenage, UK: Peter Peregrinus Ltd., 1989

[57] Kautskey J., Nichols N. K., Van Dooren P. *Robust pole-assignment in linear state feedback*. International Journal of Control, 1985, 41(5):1129–1155

[58] Dorf R. C., Bishop R. H. *Modern Control Systems*. 9th edition. Upper Saddle River, NJ: Prentice-Hall, 2001

[59] Bennett S. *Development of the PID controllers*. IEEE Control Systems Magazine, 1993, 13(2):58–65

[60] Åström K. J., Hägglund T. *PID Controllers: Theory, Design and Tuning*. Research Triangle Park: Instrument Society of America, 1995

[61] Åström K. J., Hägglund T. *Automatic Tuning of PID Controllers*. Research Triangle Park: Instrument Society of America, 1988

[62] Yu C. C. *Autotuning of PID Controllers: Relay Feedback Approach*. Advances in Industrial Control. London: Springer-Verlag, 1999

[63] Tan K. K., Wang Q.-G., Hang C. C., Hägglund T. *Advances in PID Control*. Advances in Industrial Control. London: Springer-Verlag, 2000

[64] Wang L. P., Cluett W. R. *From Plant Data to Process Control: Ideas for Process Identification and PID Design*. Research Triangle Park: Taylor & Francis, 2000

[65] Zhuang M. *Computer Aided PID Controller Design*. Ph.D. thesis, Sussex University, UK, 1992

[66] Chien K.-L., Hrones J. A., Reswick J. B. *On the automatic control of generalised passive systems*. Transactions of the ASME, 1952, 175–185

[67] Cohen G. H., Coon G. A. *Theoretical considerations of retarded control*. Transactions of the ASME, 1953, 827–834

[68] Hang C. C., Åström K. J., Ho W. K. *Refinement of the Ziegler–Nichols tuning formula*. Proceedings of the IEE, Part D, 1991, 138:111–118

[69] Wang F. S., Juang W. S., Chan C. T. *Optimal tuning of PID controllers for single and cascade control loops*. Chemical Engineering Communications, 1995, 132:15–34

[70] Zhuang M., Atherton D. P. *Automatic tuning of optimum PID controllers*. Proceedings of the IEE, Part D, 1993, 140:216–224

[71] O'Dwyer A. *Handbook of PI and PID Controller Tuning Rules*. London: Imperial College Press, 2003

[72] Visioli A. *Optimal tuning of PID controllers for integral and unstable processes*. Proceedings of the IEE, Part D, 2001, 148(2):180–184

[73] Haalman A. *Adjusting controllers for a deadtime process*. Control Engineering, 1965, 71–73

[74] McMillan G. K. *Control loop performance*. In Proceedings of the ISA/84 International Conference on Advances in Instrumentation. Houston, TX, 1984, 589–603

[75] O'Dwyer A. *PI and PID controller tuning rules for time delay processes: A summary*. Parts 1 & 2. In Proceedings of the Irish Signals and Systems Conference, 1999

[76] Nelder J. A., Mead R. *A simplex method for function minimization*. Computer Journal, 1965, 7:308–313

[77] Goldberg D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989

[78] Houck C. R., Joines J. A., Kay M. G. *A Genetic Algorithm for Function Optimization: A MATLAB Implementation*. Electronic Version of the GAOT Manual, 1995

[79] Åström K. J., Hang C. C., Persson P., Ho W. K. *Towards intellegient PID control*. Automatica, 1992, 28(1):1–9

[80] Stein G., Athans M. *The LQG/LTR procedure for multivariable feedback control design*. IEEE Transactions on Automatic Control, 1987, 32(2):105–114

[81] Zhou K. *Optimal and Robust Control*. Upper Saddle River, NJ: Prentice Hall, 1996

[82] Doyle J. C., Francis B. A., Tannerbaum A. R. *Feedback Control Theory*. New York: MacMillan Publishing Company, 1991

[83] Anderson B. D. O. *Controller design: Moving from theory to practice*. IEEE Control Systems Magazine, 1993, 13(4):16–25. Also, Bode Prize Lecture, CDC, 1992

[84] Anderson B. D. O., Liu Y. *Controller reduction: Concepts and approaches*. IEEE Transactions on Automatic Control, 1989, AC-34(8):802–812

[85] Chiang R. Y., Sofanov M. G. *Robust Control Toolbox User's Guide.* The MathWorks, Natick, MA, 1992

[86] Torvik P. J., Bagley R. L. *On the appearance of the fractional derivative in the behavior of real materials*. Transactions of the ASME, 1984, 51(4):294–298

[87] Podlubny I., Dorčák L., Misanek J. *Application of fractional-order derivatives to calculation of heat load intensity change in blast furnace walls*. Transactions of the Technical University of Kosice, 1995, 5(5):137–144

[88] Axtell M., Bise E. M. *Fractional calculus applications in control systems*. In Proceeding of the IEEE 1990 Natational Aerospace and Electronics Conference. New York, 1990, 563–566

[89] Dorčák L. *Numerical models for simulation the fractional-order control systems*. UEF SAV, The Academy of Sciences Institute of Experimental Physics. Kosice, Slovak Republic, 1994, 62–68

[90] Matignon D. *Stability result on fractional differential equations with applications to control processing*. In IMACS-SMC Proceedings. Lille, France, 1996, 963–968

[91] Oldham K. B., Spanier J. *The Fractional Calculus*. New York: Academic Press, 1974

[92] Podlubny I. *The Laplace transform method for linear differential equations of the fractional order*. In Proceedings of the 9th International BERG Conference. Kosice, Slovak Republic, 1997, 119–119 (in Slovak)

[93] Podlubny I. *Fractional Differential Equations*. San Diego: Academic Press, 1999

[94] Woon S. C. *Analytic continuation of operators — operators acting complex s-times. Applications: from number theory and group theory to quantum field and string theories*. Reviews in Mathematical Physics, 1999, 11(4):463–501

[95] Zavada P. *Operator of fractional derivative in the complex plane*. Communications in Mathematical Physics, 1998, 192(2):261–285

[96] Oustaloup A. *La Dérivation non Entière*. Paris: HERMES, 1995

[97] Petráš I., Dorčák L., Kostial I. *Control quality enhancement by fractional order controllers*. Acta Montanistica Slovaca, 1998, 2:143–148

[98] Podlubny I. *Fractional-Order Systems and Fractional-Order Controllers*. Technical Report UEF-03-94, The Academy of Sciences Institute of Experimental Physics, Kosice, Slovak Republic, 1994

[99] Podlubny I. *Fractional-order systems and $PI^{\lambda}D^{\mu}$-controllers*. IEEE Transactions on Automatic Control, 1999, 44(1):208–214

[100] Magin R. L. *Fractional Calculus in Bioengineering*. Redding, CT: Begell House Publishers, 2006

[101] Miller K. S., Ross B. *An Introduction to the Fractional Calculus and Fractional Differential Equations*. New York: Wiley, 1993

[102] Samko S. G., Kilbas A. A., Marichev O. I. *Fractional Integrals and Derivatives and Some of Their Applications*. Minsk: Nauka i Technika, 1987

[103] Xue D., Chen Y. Q. *MATLAB Solutions to Advanced Applied Mathematical Problems*. Beijing: Tsinghua University Press, 2004. (in Chinese)

[104] Hilfer R. *Applications of Fractional Calculus in Physics*. Singapore: World Scientific, 2000

[105] Petráš I., Podlubny I., O'Leary P. *Analogue Realization of Fractional Order Controllers*. Fakulta BERG, TU Košice, 2002

[106] Oustaloup A., Levron F., Mathiew B., Nanot F. *Frequency band complex noninteger differentiator: Characterization and synthesis*. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2000, 47(1):25–39

[107] Xue D., Zhao C. N., Chen Y. Q. *A modified approximation method of fractional order system*. In Proceedings of the IEEE Conference on Mechatronics and Automation. Luoyang, China, 2006, 1043–1048

[108] Åström K. J. *Introduction to Stochastic Control Theory*. London: Academic Press, 1970

[109] Xue D., Zhao C. N., Chen Y. Q. *Fractional order PID control of a DC-motor with elastic shaft: A case study*. In Proceedings of the American Control Conference. Minneapolis, MN, 2006, 3182–3187

[110] Xue D., Goucem A., Atherton D. P. *A menu-driven interface to PC-MATLAB for a first course on control systems*. International Journal of Electrical Engineering Education, 1991, 28(1):21–33

Index of MATLAB Functions

Bold page numbers indicate where to find the syntax explanation of the function

Index