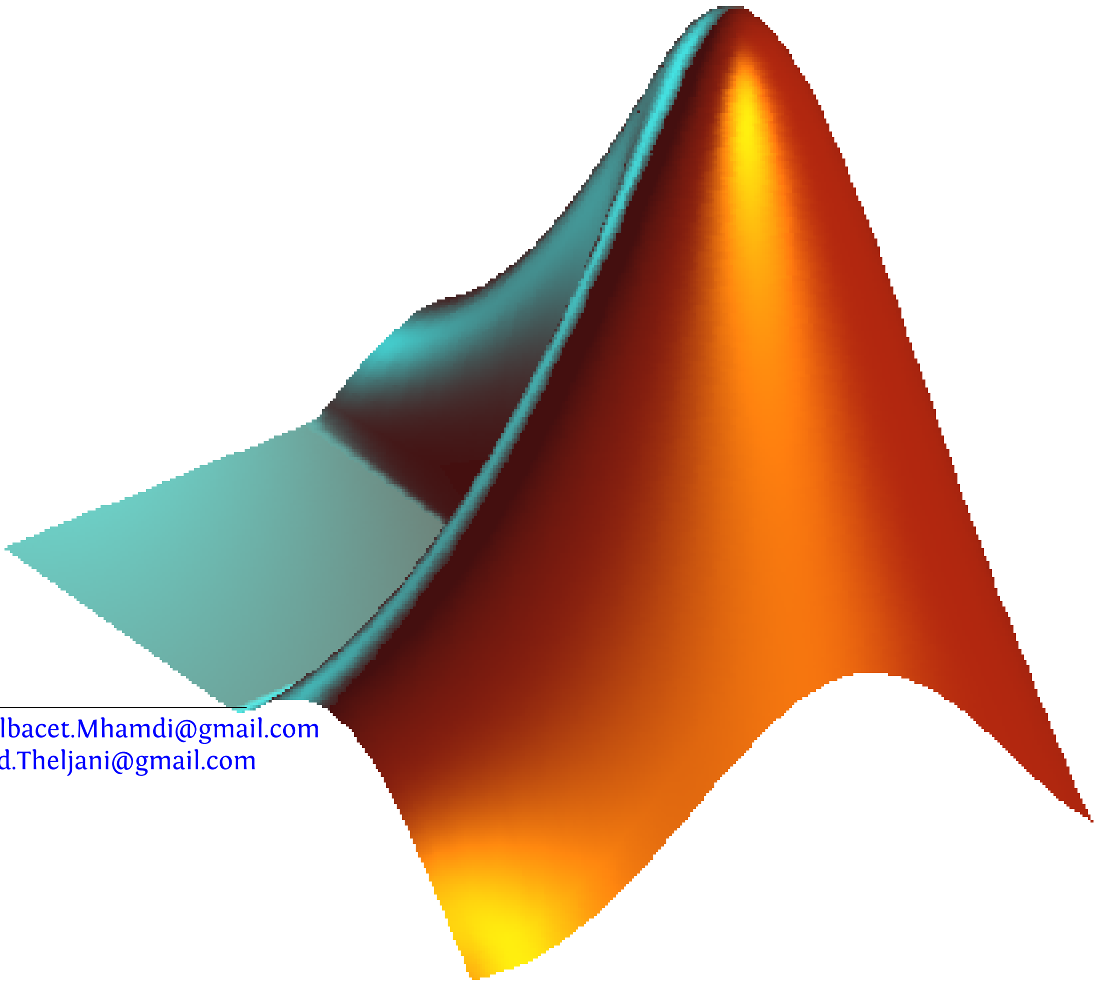
Mathematical Analysis & System Control with Matlab

PREPARED BY: **Abdelbacet Mhamdi**[1] DR.-ING. IN ELECTRICAL ENGINEERING
Foued Theljani[2] DR IN ELECTRICAL ENGINEERING

Automatic Control and System Engineering



[1]e-Mail: Abdelbacet.Mhamdi@gmail.com
[2]e-Mail: Foued.Theljani@gmail.com

Foreword

THIS MANUAL is intended for engineers who want to start with Matlab. We took Automatic Control as a field of application. Some examples have been discussed in this document. Reader can test them on his machine. Just, follow the steps detailed in each section.

We tried to make numerous references for further reading. They are available, either in hard copy or electronic. Students can benefit from sources in order to enlarge their capabilities in modeling, designing and analysis of physical situations.

We have treated all exercises with particular care. Most cases, the mathematical equations used in this manual refers to a well-known problem like ”Fibonacci equation”, ”Legendre polynomials”, etc. And, sometimes, they were taken as simple demonstrations, they may or not have any physical meaning. They are most commonly used in case of syntax training.

Some parts may not be well explained. This document is a draft version, we assume. More explanations and definitions are going to be held in earlier versions.

Please, contact us in case you find an error or mysterious ideas. The more you criticize, better becomes the document.

The URLs shown in footnotes are indicatives. They have been visited in 2013. We do not guarantee their validities, and we are not responsible on updates mis–leaded by some websites.

ACKNOWLEDGMENT

To whom all thanks should be addressed for the linguistic verification: ***, english professor at ***.

Table of Contents

CHAPTER 1

Getting Started

There are things which seem incredible to most men who have not studied Mathematics.

Archimedes of Syracus

Contents

We will present, in this chapter, a brief introduction about Matlab. We will also show how to install it on a hard disk and exhibit the reasons behind writing this manual.

1.1 What is Matlab ?

Matlab, developed by Mathworks, is a numerical computing environment. It allows data manipulations, implementation of algorithms, plotting functions and creating Graphical User Interfaces (GUI). One of the most major features of Matlab is the interfacing capabilities with programs written in other languages, including C, C^{++}, Java, and Fortran, and with others Integrated Development Environment (IDE) such that LabVIEW[1] or Integrated Software Environment (ISE) like XILINX[2] and even simulator like PSIM[3].

The conclusion is that Matlab is present as a third-party product where there is a matrix manipulations due to its acronym: Matrix Laboratory.

Matlab users come from various domains of engineering, economics and science. It is widely used in industry, academic institutions as well as research structures.

After installing Matlab, you will find a dozens of toolboxes for images acquisition, signal processing, designing control laws...that have been developed by experts of The MathWorks.

1.2 How to install Matlab ?

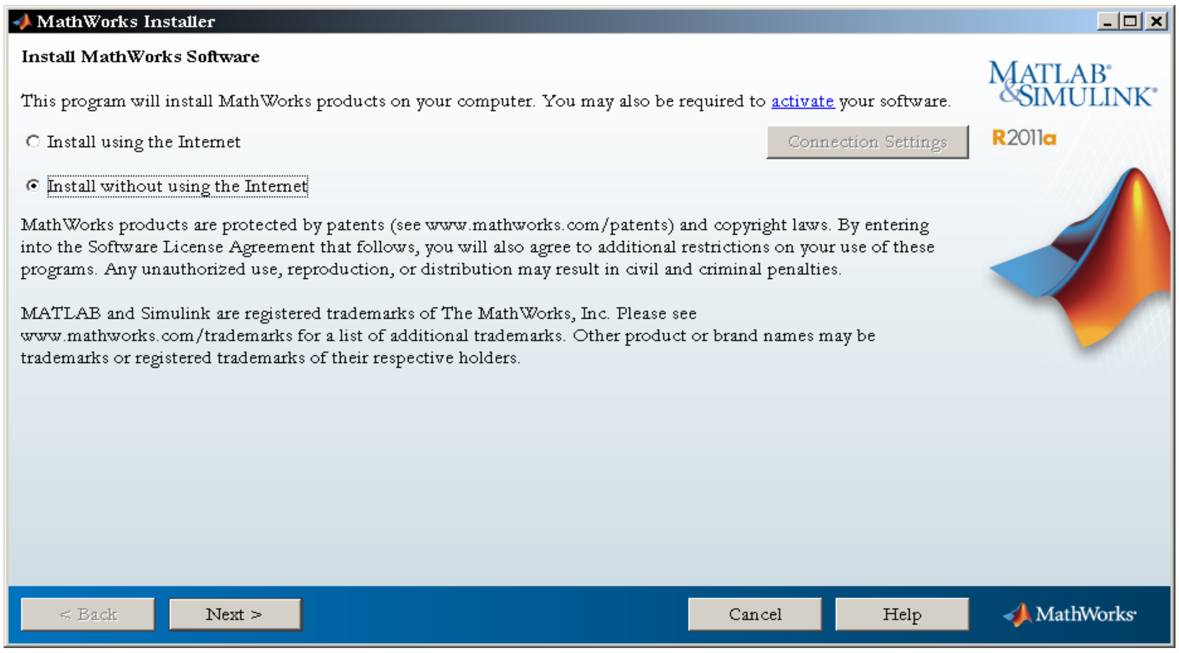
In this section, we will show how to install Matlab.

In the setup folder, launch setup. the installer will automatically choose between your OS computer version (32 bits or 64 bits). You will get on your screen the figure 1.1. Accept the license agreement and click Next. Check the box "I have the File Installation Key for my license" and enter then the key. it looks like a series of digits.

Contact your instructor to get a valid key or purchase one directly from the official website of Mathworks[4].

Let the box typical checked and click next. Specify then the installation folder and clikc next. I recommend to let them default. You will get, figure 1.2 the fully compatibles libraries and toolboxes licensed on your machines. You can take benefits from them. Click Next to pursue the installation. Figure 1.3 shows the slide bar of installation. Wait until Matlab finish installation. Ultimately, activate your version. For this reason, choose activate without using Internet and then click on browse as shown on figure 1.4 and give the full path to your file, it has the extension "*.dat". You, now, have a fully functional version of Matlab. I recommend you to launch it and search for demos, nothing else.

[1]http://www.ni.com/labview/
[2]http://www.xilinx.com/
[3]http://www.psim-europe.com/
[4]For purchase: http://www.mathworks.com/company/aboutus/contact_us/contact_



This program will install MathWorks products on your computer. You may also be required to activate your software.

Install using the Internet

[✓ Install without using the Internet]

MathWorks products are protected by patents (see www.mathworks.com/patents) and copyright laws. By entering into the Software License Agreement that follows, you will also agree to additional restrictions on your use of these programs. Any unauthorized use, reproduction, or distribution may result in civil and criminal penalties.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. Please see www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Figure 1.1: Matlab setup

1.3 Matlab desktop

1.3.1 Desktop layout

Like any other environment, we dispose in Matlab a menu bar and a tool bar where user can find the major commands needs to configure the preferences of Matlab, this means how to let Matlab behaves. Then, the main interface is split into four majors areas as shown in figure 1.5.

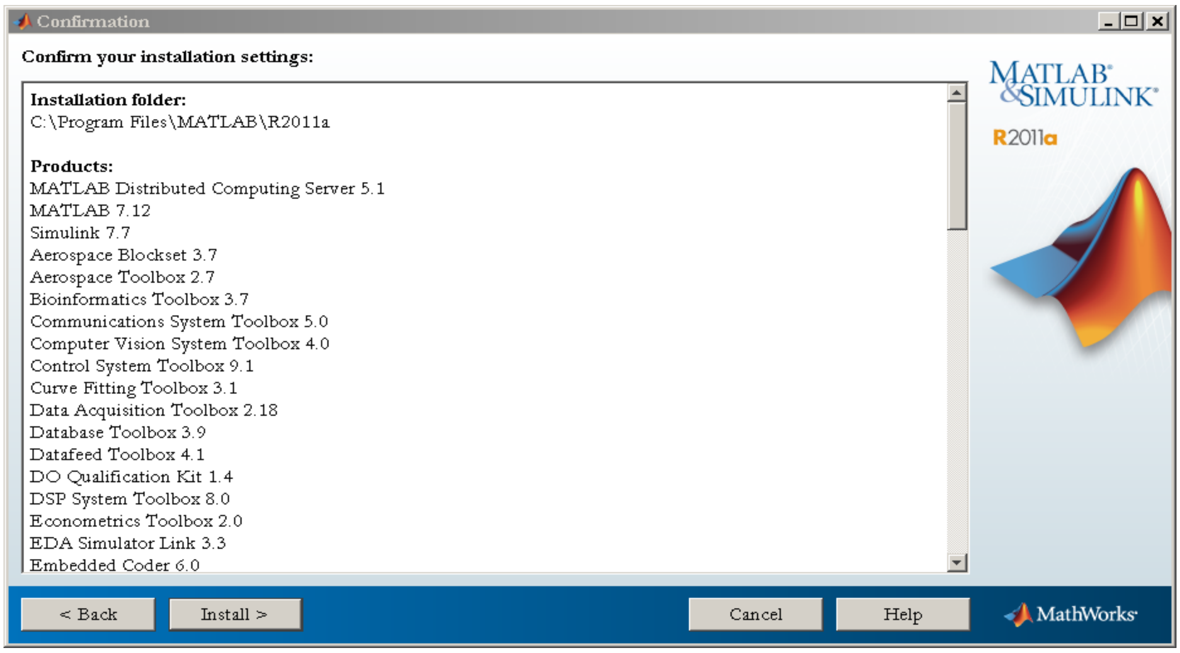
a) Area 1

it is the Command Window, where user can strike the Matlab commands and see results displayed on the same screen.

b) Area 2

It is the Workspace. All variables are saved in this area. If you need to know more about saved variables, just try this command on area 1:

sales.html



Confirm your installation settings:

Installation folder:
C:\Program Files\MATLAB\R2011a

Products:
MATLAB Distributed Computing Server 5.1
MATLAB 7.12
Simulink 7.7
Aerospace Blockset 3.7
Aerospace Toolbox 2.7
Bioinformatics Toolbox 3.7
Communications System Toolbox 5.0
Computer Vision System Toolbox 4.0
Control System Toolbox 9.1
Curve Fitting Toolbox 3.1
Data Acquisition Toolbox 2.18
Database Toolbox 3.9
Datafeed Toolbox 4.1
DO Qualification Kit 1.4
DSP System Toolbox 8.0
Econometrics Toolbox 2.0
EDA Simulator Link 3.3
Embedded Coder 6.0

Figure 1.2: List of installed products

Code

```
>> whos
```

The results are the names of variables, the min and max values for every variables, class and the number of bytes needed to save it.

c) Area 3

It is the Current Folder, the path indicated by Matlab to execute a particular code. if you need to execute an algorithm which is not on the path shown by are 4, you have then to change the folder. Otherwise, an error message will be displayed on command window if it is not the absolute path.

d) Area 4

It is the Command History, where all instructions are saved in a panel from the date of installing Matlab on your machine until now.

1.3.2 How to personalize the desktop layout

In the menu bar, click on Desktop, Desktop Layout then load your preferences. For example, here the customized layout was saved under the name MyLayout.

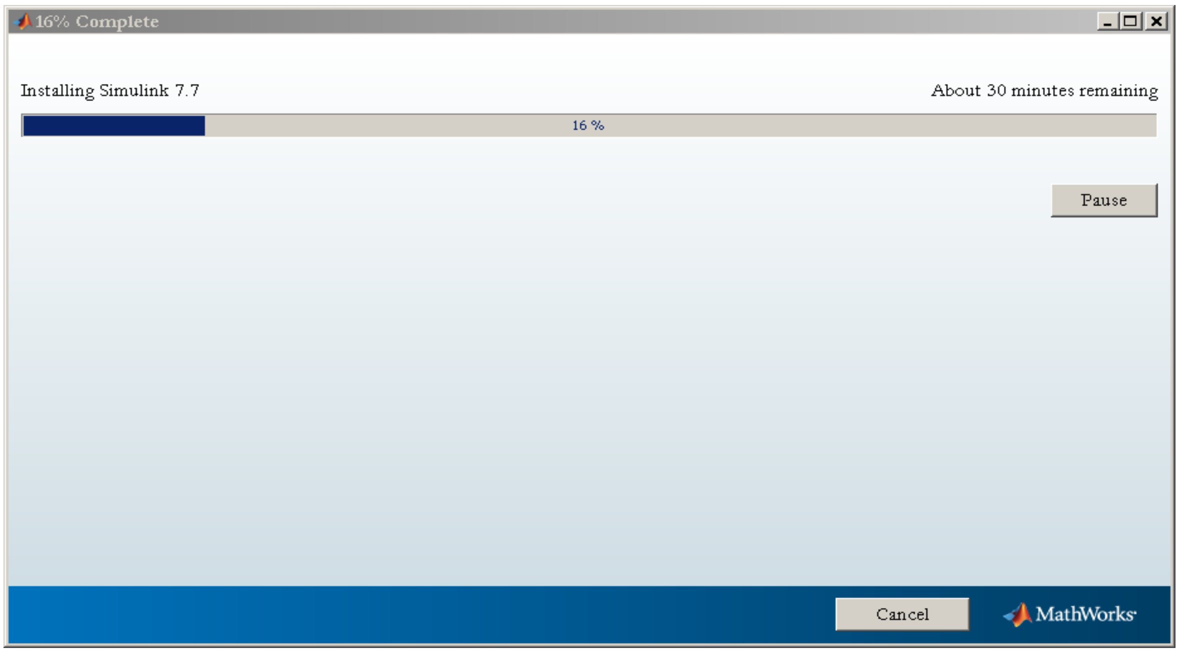


Figure 1.3: Slide bar of installation

You just click on it, and your layout appears like the display shown on figure 1.6.

1.4 Why Matlab ?

Matlab is an indispensable tool for engineers. The immediate reason is because of all what you need in Matlab is the basics pieces of informations needed to solve the problem. There is no need to configure your environment to adopt your algorithm, just know a little bit about the syntax.

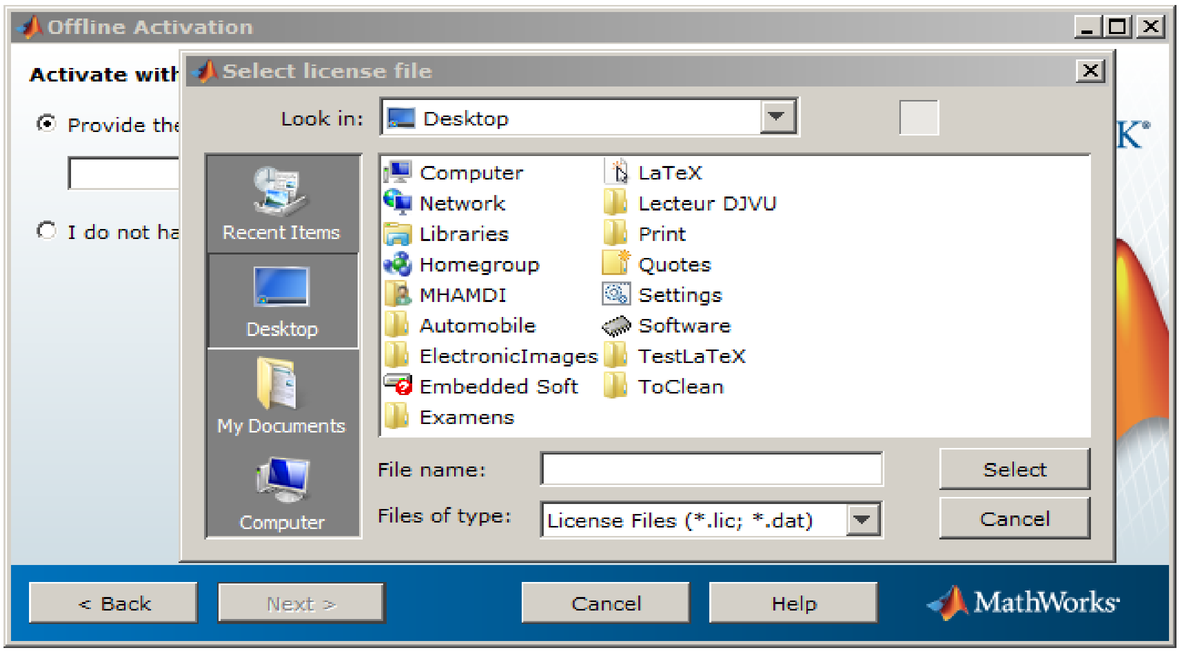
Matlab is an easy to use environment, it is a fourth-generation programming language[5] (4GL). It is a very high level language.

For a full list of Matlab's users stories, follow this link to watch videos[6]: http://www.mathworks.com/company/user_stories/index.html

1.5 Conclusion

Matlab is an interactive matrix calculator, with a full-fledged programming environment. Matlab as a language is extremely frustrating to handle with

[5]http://en.wikipedia.org/wiki/Fourth-generation_programming_language
[6]Think of using matlab to launch the integrated web browser. Try » web



Look in: Desktop

Computer, Network, Libraries, Homegroup, MHAMDI, Automobile, ElectronicImages, Embedded Soft, Examens, LaTeX, Lecteur DJVU, Print, Quotes, Settings, Software, TestLaTeX, ToClean

File name:

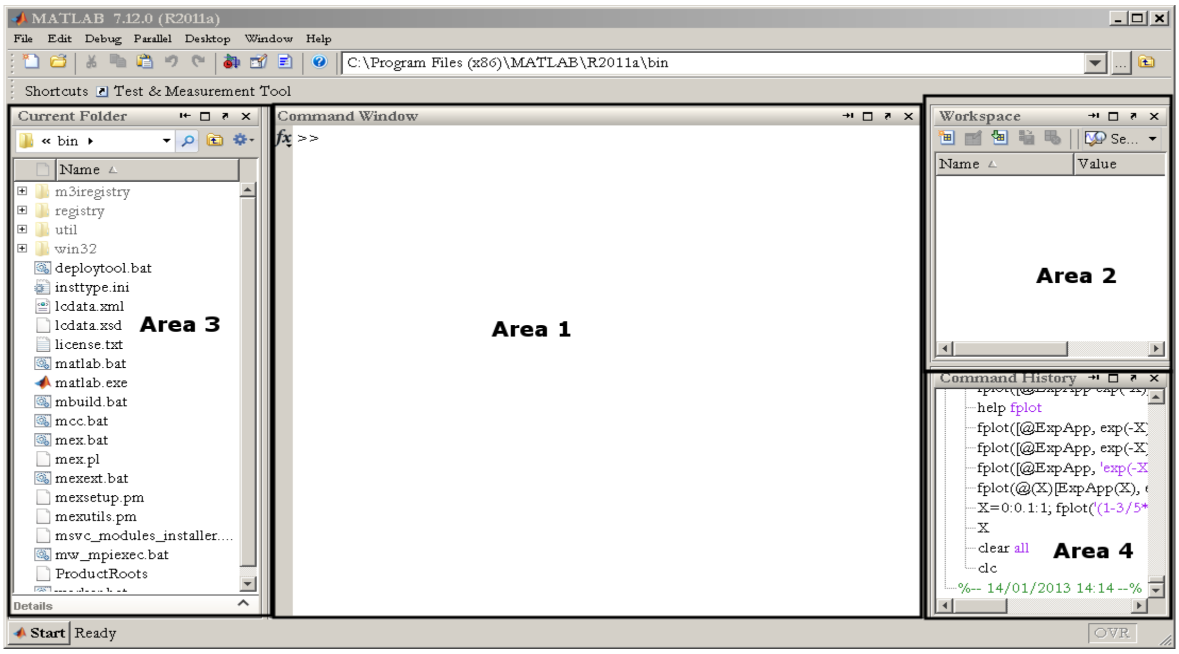
Files of type: License Files (*.lic; *.dat)

Select

Cancel

Figure 1.4: Matlab's activation

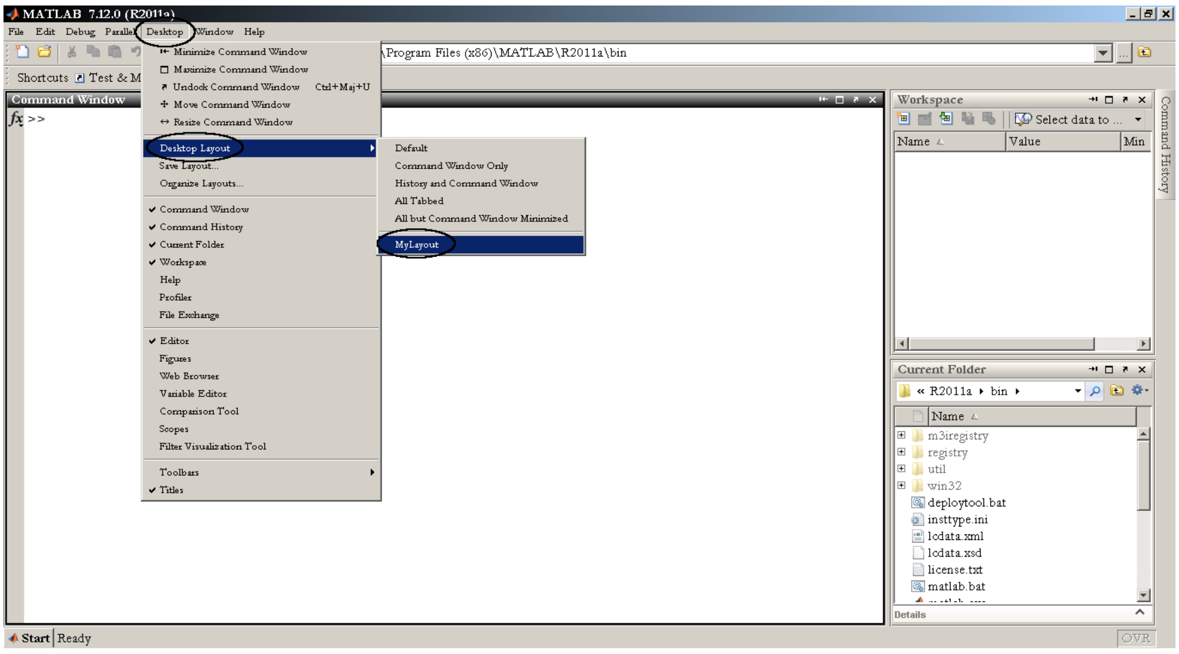
general tasks and very delicate purpose. Your focus with Matlab is what you need to program not how to program it.



```
>>
```

```
help fplot
fplot([@ExpApp, exp(-X
fplot([@ExpApp, exp(-X
fplot([@ExpApp, 'exp(-X
fplot(@(X)[ExpApp(X), e
X=0:0.1:1; fplot('(1-3/5*
X
clear all
clc
%-- 14/01/2013 14:14 --%
```

Figure 1.5: Matlab layout



```
>>
```

Figure 1.6: Desktop layout

CHAPTER 2

GUI: Graphical User Interface

Mathematics is the art of giving the same name to different things.

Henri Poincaré

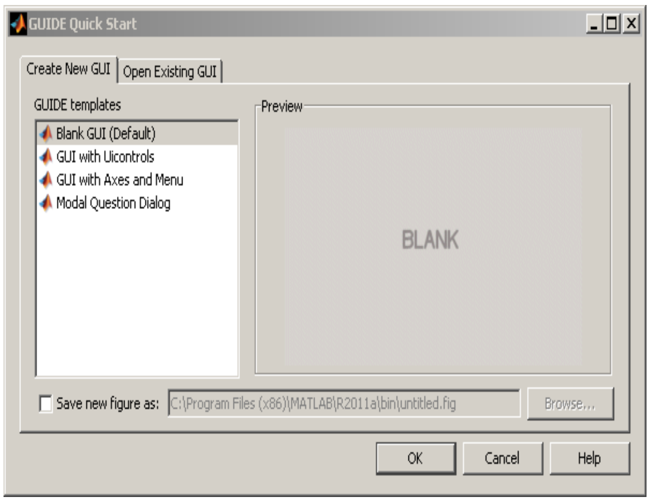
Contents

MATLAB is not just an environment for technical computing, where we can solve equations. It is also an environment of graphical interfaces development. This integrated software is called "GUIDE", Graphical User Interface Development Environment [5, 6].

In this chapter, we will take an example of a simple basic calculator in order ta have a close look on how to prepare a GUI. For this reason, try, at the command window, the instruction:

Code
```
>> guide
```

or simply move your mouse on the toolbar, you will have a shortcut like the icon . Just click on it, you will then launch the environment of graphics. You will get on your screen, the dialog shown in figure 2.1. You can access the templates or simply click next.


GUIDE Quick Start

Create New GUI | Open Existing GUI

GUIDE templates
- Blank GUI (Default)
- GUI with Uicontrols
- GUI with Axes and Menu
- Modal Question Dialog

Preview

Save new figure as: C:\Program Files (x86)\MATLAB\R2011a\bin\untitled.fig

Figure 2.1: Guide quick start

You will be well oriented by your instructor, just follow steps and ask questions.

Now, the environment is ready to use, as in figure 2.2. Save your work, you will notice the creation of two files on you directory with the same name, but just with two different extensions, one has this termination *.fig an the second one is an m-file.

In the coming sections, we will evoke the purposes of each file.

2.1 Fig-File

The Fig-file contains the graphical declaration or the instantiation of the all used graphical objects in your interface. The main GUI that appears to user in order to customize his interface contains a palette where you can easily access the preconfigured objects like ”pushbutton”, ”toggle button” and ”slider”. You can even add some external objects via the activeX control which is not the object of this chapter. Try to put the objects needs as in figure 2.3. Notice that

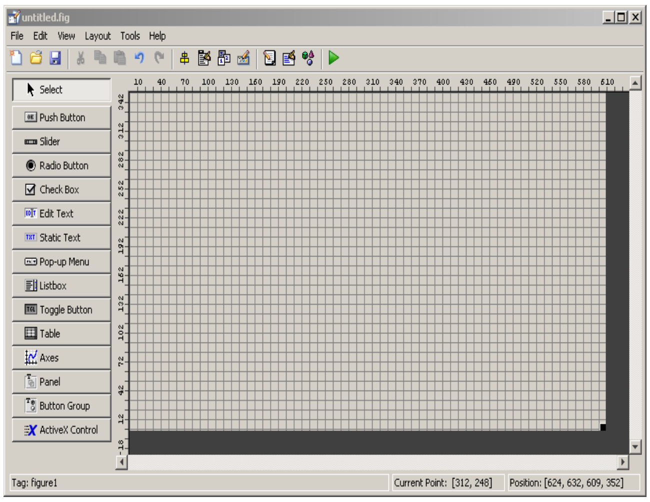


Figure 2.2: Guide: Main GUI

you have for any object an Object Browser which, when pointed on an object, list all properties relative to this one and even the set of methods called when evoking it. You may also need to know that there exist an uipanel, use it in case of you need to regroup elements by similarities.

Now, you can launch your application. Your conceived interface may look like this one indicated in figure 2.4. We notice that even when we push on a button or check a box, nothing happens. This is due to the nature of the file itself. It is a fig-file, it contains only the calculator interface instantiation, which means how does it look on screen when we launch it. The behavior of the application itself is stored in the m-file.

2.2 M-File

The idea of the m-file is to associate to each object a function which acts only if it is involved. For instance, user needs to make an operation between two numbers, so under the callback associated to edit text, returned variables have to be saved in numeric double variable. In order to save the entered value in a variable, you can use the following syntax.

Code
```
b = str2double(get(hObject,'String'));
% hObject is a handle for the current object.
```

It may for some mis-manipulation, that user enter a text in the field of digits, so, programmer has to generate an initialization code and generate an error dialog.

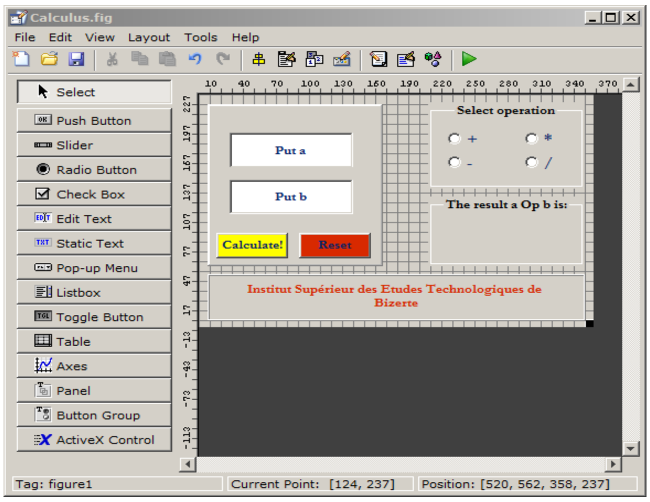


Figure 2.3: The calculator figure

Code
```
if isnan(b) % Test if b is not a numeric
    % Show a zero in the edit text
    set(hObject, 'String', 0);
    % Generate an error dialog
    errordlg('Input must be a number','Error');
end
```

Before, quitting the function, you have to update the GUI and save all variable in the general structure with handles and user data[1].

Code
```
handles.Data.b=b;
guidata(hObject, handles); % Update handles structure
```

The ultimate application will behave like the example displayed on figure 2.5. Enter your digits, choose the desired operation and then just click Calculate. The result will be displayed on right half of the interface.

2.3 Conclusion

We have seen just a small glimpse on developing graphical user interfaces, for more sophisticated, you can access the website of mathworks. In the next chapters, we are going to exhibit the reasons for calling it the Matrix Laboratory.

[1]Visit: http://www.mathworks.com/help/matlab/gui-building-basics.html

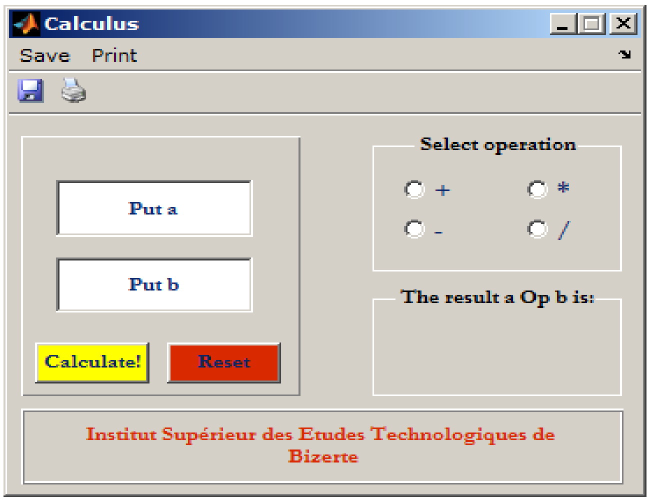
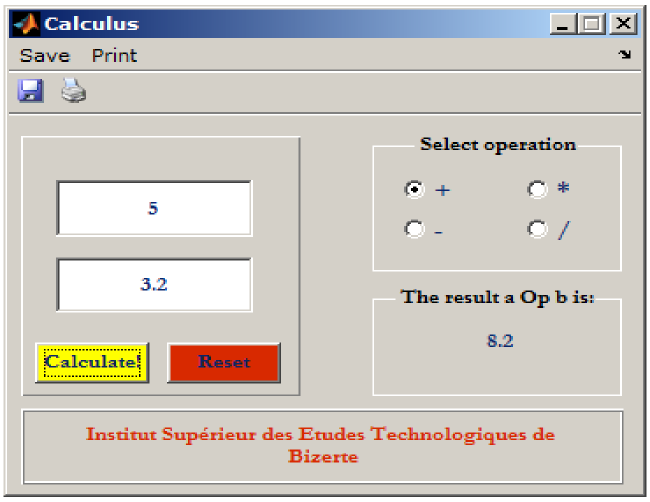


Figure 2.4: The calculator application



5

3.2

+ [✓]

8.2

Figure 2.5: The calculator application: Demonstration

CHAPTER 3

Linear Algebra

Perfect numbers like perfect men
are very rare.

Rene Descartes

Contents

BENEATH the surface of the world, it exists other rules of science but beneath them, there is a far deeper such a rule, a matrix of pure mathematics which explains the nature of the rule of science and how we can understand them in the first place[1].

Applications cases of algebra are real and efficient but we do not make a prior conditioning for them. Instead, for some applications which need more accuracy and particular care, we may use a paper and a pen to solve the puzzles.

This chapter will show the interest of Matlab as a tool for resolving the previous evoked problems. You don't need to know much about algebra theory with it all complicated demonstrations and properties, passes the arguments for Matlab and wait for result.

3.1 Matrix definition

A matrix with n rows and m columns is a rectangular table of nxm elements arranged. There is n rows in the matrix M, and every row has m elements.

For example, consider the following matrix M, with complex coefficients.

$$M = \begin{pmatrix} 0 & 1 & -3 & 2\jmath & 56 & 26 \\ 2 & 4 & 3 & 65 & -14 & 0 \\ 1 & 12 & 4.44 & 7 & -3.5\jmath & 0 \\ 6 & 0 & 9 & 9 & 3.15 & 67 \end{pmatrix}$$

[1]Documentary film: "Dangerous Knowledge", presented by DAVID MALONE, I-330 Limited for BBC Channel.

The matrix M has the dimension $(4,\ 6)$. In this representation, the first number gives the number of rows, the second one informs on columns' number. To spot out an element in a matrix, we indicate the row's index followed by the index of the column. Rows are enumerated from above to below and columns from left to right.

We note thereafter by $m_{(i,\,j)}$ the element in the matrix M, located at the i^{th} row and the j^{th} column. For example, $m_{(3,\,5)} = -3.5$j

Here, we make some definitions of special matrices.

- Square matrix: $n = m$ (The matrix has as rows as columns number.)
- Row matrix: $m = 1$ (Number of columns is equal to one.)
- Column matrix:$n = 1$ (Number of rows is equal to one.)

Generally, the arrangement of elements in matrix M, of dimension $(m,\ n)$ is as in equation 3.1:

$$M = \begin{pmatrix} m_{(1,\,1)} & \cdots & \cdots & \cdots & m_{(1,\,m)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & m_{(i,\,j)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ m_{(n,\,1)} & \cdots & \cdots & \cdots & m_{(n,\,m)} \end{pmatrix} \tag{3.1}$$

3.2 Operations on matrices

3.2.1 The transpose

Assume that M_1 is a matrix of dimensions $(n,\ m)$, the transpose of M_1 is a also a matrix M_2 having the dimension $(m,\ n)$ such that the coefficient $m_{2_{(i,\,j)}} = m_{1_{(j,\,i)}}$ with $(i,\ j) \in \{1\cdots n\} \times \{1\cdots m\}$. We note by $M_2 = M_1^T$.

3.2.2 Sum of two matrices

Given two matrices A and B of size $(n,\ m)$, the matrix sum of $A + B$ is a new matrix C with:

$$C = A + B;\ c_{(i,\,j)} = a_{(i,\,j)} + b_{(i,\,j)} \text{ where } (i,\ j) \in \{1..n\} \times \{1..m\} \tag{3.2}$$

3.2.3 Product of two matrices

Assume that A and B are two matrices of following dimensions: (n_A, m_A) and (n_B, m_B). The sufficient condition to compute the matrix product $C = A \times B$ is that $m_A = n_B = p$. C is given by equation 3.3.

$$c_{ij} = \sum_{k=1}^{p} a_{(i,k)} b_{(k,j)} \tag{3.3}$$

The matrix C has a dimension of (n_A, m_B), which means that C has as many rows as A and as many columns as B.

3.2.4 Base changing

Assume that:

$$\begin{aligned} f : \mathbb{C}^p &\to \mathbb{C}^l \\ \overrightarrow{u} &\longmapsto \overrightarrow{v} = f(\overrightarrow{u}) \end{aligned}$$

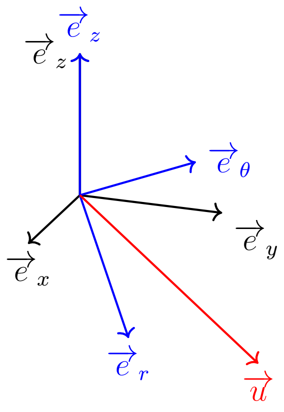
Consider that the starting space is regularized by a base called b_1, which contains p non zero and linearly independents elements. The space $\mathbb{C}^l$ is regularized by the base B_2, it itself contains l non zero and linearly independent elements. We can then associate to the function handle f a matrix M. M can be seen as an arrangement of relationships between the two bases B_1 and B_2 with respect to the function f. It is, clearly, of dimension (p, l).

$$\overrightarrow{v} = f(\overrightarrow{u}) = M\overrightarrow{u} \tag{3.4}$$

The vector $\overrightarrow{v}$ is then the image of vector $\overrightarrow{u}$ by the function f given that the description of a vector is always relative to the base where the vector is written.

Changing the base influences the description of the vector. This movement from one base to another can be always translated by a multiplication.

Consider this example:



$$\overrightarrow{u}_{B_2} = r\overrightarrow{e}_r + z\overrightarrow{e_z}$$

$$\overrightarrow{u}_{B_1} = P\overrightarrow{u}_{B_2}$$

$$\overrightarrow{u}_{B_1} = P\begin{pmatrix} r \\ 0 \\ z \end{pmatrix}_{B_1}$$

$B_1 = (\overrightarrow{e}_x, \overrightarrow{e}_y, \overrightarrow{e}_z)$ and $B_1 = (\overrightarrow{e}_r, \overrightarrow{e}_\theta, \overrightarrow{e}_z)$ are respectively the direct Cartesian and cylindric orthonormal basis of $\mathbb{R}^3$.

Assume that $\overrightarrow{u}$ is a vector in the 3D space, so, $\overrightarrow{u}$ can be written in these following descriptions.

$$\overrightarrow{u} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{B_1} = \begin{pmatrix} r \\ 0 \\ z \end{pmatrix}_{B_2}$$

We note by P the passage matrix from the base B_2 to the baseB_1, so, we have to write the elements in base B_1 according to those in the base B_2.

$$P = P_{B_2 \to B_1} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Then, any vector written in the cylindric referential can be transformed to the Cartesian base, just by multiplication of the matrix P.

$$\overrightarrow{u}_{B_1} = \begin{pmatrix} r\cos(\theta) \\ r\sin(\theta) \\ z \end{pmatrix}_{B_1}$$

We will get the following system of equations:

$$\begin{cases} x & = r\cos(\theta) \\ y & = r\sin(\theta) \\ z & = z \end{cases}$$

3.2.5 Trace of a matrix

Assume that M is a square matrix, with complex coefficients of dimension n. We call trace of M the sum of the elements which are located on the diagonal, i.e, the elements having identical index (i.e. $i = j$ with $i, j \in \{1 \cdots n\}$).

$$Trace(M) = \sum_{k=1}^{n} m_{(k,k)} \tag{3.5}$$

3.2.6 Determinant of a matrix

To compute the determinat of a square matrix, we can proceed in two different ways (Caution: Not both of them). We choose the row or the column that conatins the maximum of zeros. The determinant can be computed as in equation 3.6.

$$det(M) = \sum_{i(respec.\,j)=1}^{n} (-1)^{i+j} m_{(i,j)} \triangle_{ij} \tag{3.6}$$

$\triangle_{ij}$ is the determinant of the matrix M after deleting the i^{th} row and j^{th} column.

3.2.7 Eigenvalues and Eigenvectors

We call an eigenvalue of matrix M, the coefficient λ who verifies the property 3.7:

$$M\overrightarrow{\psi} = \lambda\overrightarrow{\psi}, \tag{3.7}$$

The vector $\overrightarrow{\psi}$ who verifies the previous equation is called the eigenvector associated to the eigenvalue λ. It may exist more than one eigenvector, for whom the statement previously made is checked. The number of these linearly independent vectors is called the multiplicity of λ.

If one eigenvalue λ at least has geometric multiplicity larger than its algebraic multiplicity, then the set of linearly independent eigenvectors of M can not form a basis of $\mathbb{R}^n$, thereby implying that matrix M fails to be diagonalizable.

3.2.8 Generalized eigenvector

Consider an $n \times n$ matrix M. A generalized eigenvector for M is a vector ψ for which

$$(M - \lambda I)^k \psi \;=\; 0, \tag{3.8}$$

where k is a positive integer ($k \in \mathbb{Z}^*$). I denotes here the identity matrix. The scalar λ is called the generalized eigenvalue associated to ψ. An eigenvector is a generalized eigenvector corresponding to $k = 1$. We call generalized eigenspace for λ the linear span of all ψ associated to some generalized eigenvalue λ.

When a square matrix M fails to be diagonalizable, the generalized eigenvectors appear to be of particular interest. The main aim of determining the generalized eigenvector is to enlarge the set of linearly independent eigenvectors in order to form a basis for $\mathbb{R}^n$.

3.2.9 The spectrum of a matrix

We call the spectrum of a square matrix, the set of all her eigenvalues with their multiplicities orders.

? How to compute the spectrum of a square matrix ?

$Sp(M) = \{\lambda \epsilon \mathbb{C} \backslash \chi(M) = 0\}$, $\chi(M)$ is called the characteristic polynomial of the matrix M, with $\chi(M) = det(\lambda I_n - M)$.

Remark 1

If the cardinal of the spectrum of a matrix is equal to her size, then we can diagonalize this matrix.

We call the resolvent of a square matrix A, the term $(sI_n - A)^{-1}$. it can be written as:

$$(sI_n - A)^{-1} = \frac{adj(sI_n - A)}{det(sI_n - A)}. \tag{3.9}$$

$adj(sI_n - A)$ is given in equation 3.10:

$$adj(sI_n - A) = \sum_{i=0}^{n} R_i s^{n-i} \tag{3.10}$$

The approach for computing the coefficients R_i and I_i is given according to the algorithm of Leverrier-Souriau-Frame-Faddeev.

$R_i = R_{i-1}A + d_{i-1}I_n, \quad d_i = -trace(R_iA),$ where $i \in \{1, \cdots, n\}$
and $R_1 = I_n$, $d_1 = -trace(R_1A)$

3.2.10 Rank of a matrix

The rank of a matrix is the number of linearly independent rows or columns.

3.2.11 The inverse of a matrix

Given a square matrix A of size n. A is called reversible if and only if its determinant is different of zero. We call B the inverse matrix of A, the matrix which verifies the following equality: $A \times B = B \times A = I_n$, I_n is called the identity matrix. We note this matrix by A^{-1}.

Direct method The inverse of a non singular matrix A ($det(A) \neq 0$) is computed according to equation 3.11.

$$A^{-1} = \frac{1}{det(A)} com_A^T, \tag{3.11}$$

where com_A denotes the comatrix of A . We assume $B = com_A$; then, $b_{(i,j)} = (-1)^{i+j} \times \triangle_{ij}$. $\triangle_{ij}$ is the determinant of the matrix M after deleting the i^{th} row and j^{th} column.

Pivot method of Gauss (A brief description) It consists on performing operations on rows and columns until reaching the identity matrix, then repeat again the reverse process on the identity matrix.

Exercise 1

1. Create two matrices A and B[a]: $A = \begin{pmatrix} 1 & 2 \\ 4 & -1 \end{pmatrix}, B = \begin{pmatrix} 4 & -2 \\ -6 & 3 \end{pmatrix}$.
2. Compute $C_1 = A + B$ and $C_2 = A - B$.
3. Compute the matrix products $D_1 = AB$ and $D_2 = BA$.
4. Using element by element operations, compute the matrix F whose elements are obtained as follows: $f_{ij} = b_{ij} + a_{ij} b_{ij}^{\frac{1}{3}}$.
5. Are A and B singular? If no, compute their inverse.
6. Compute the eigenvalues of B.
7. In A, substract to the second row, the first row multiplied by 3.

[a]Visit: http://www.mathworks.com/help/matlab/matrices-and-arrays.html

Code

```
% Question 1
>> A = [1, 2; 4, -1]; B = [4, -2; -6, 3];
% Question 2
>> C1 = A+B; C2 = A-B;
% Question 3
>> D1 = A*B; D2 = B*A;
% Question 4
>> F = B+A.*(B.^(1/3));
% Question 5, obviously, B is singular
>> det(A); det(B);
>> inv(A);
```

```
% Question 6, results will be stored in the variable ans.
>> eig(B);
% Question 7
>> A(2, :) = A(2, :)-3*A(1, :);
```

Exercise 2

Given $X = [4,\ 1,\ 6]$ and $Y = [6,\ 2,\ 7]$, compute the following arrays:

1. The matrix A whose elements are $a_{ij} = x_i y_j$.
2. The matrix B whose elements are $b_{ij} = \frac{x_i}{y_j}$.
3. The vector C whose elements are $c_i = x_i y_{3-i}$.

Code

```
>> X = [4, 1, 6]; Y = [6, 2, 7];
% Question 1
>> A = X'*Y;
% Question 2
>> B = X'*(1./Y);
% Question 3
>> for i=1:3,
C(i) = X(i)*Y(4-i); % Index of a matrix can not be equal to zero.
end
```

Exercise 3

Given $X = \begin{pmatrix} 5 & 0.35 & -3.5 & 5.47 & -2 \end{pmatrix}$, what are the commands that will execute the following operations:

1. Set the negative values of X to zero.
2. Extract the values of X greater than 3 in a vector Y.
3. Add 3 to the values of X that are even.
4. Set the values of X that are less than the mean to zero.
5. Set the values of X that are greater than the mean to their difference with the mean.

Code
```
>> X = [5, 0.35, -3.5, 5.47, -2];
% Question 1, use a temporary structure...
% Vector X should always remain intact
>> Prov = X; Prov(Prov<0)=0;
% Question 2
>> Y = X(X>3);
% Question 3
>> Prov = X; Prov(rem(Prov, 2)==0) = Prov(rem(Prov, 2)==0)+3;
% Question 4
>> Prov = X; Prov(Prov<mean(X) = 0;
% Question 5
>> Prov = X; Prov(Prov>mean(X)) = Prov(Prov>mean(X))-mean(X);
```

Exercise 4

Assume that $X = \begin{pmatrix} 3 \\ 2 \\ 6 \\ 8 \end{pmatrix}$ and $Y = \begin{pmatrix} 4 \\ 1 \\ 3 \\ 5 \end{pmatrix}$.

1. Add the sum of the elements in X to Y.
2. Raise each element of X to the power specified by the corresponding element in Y.
3. Divide each element of Y by the corresponding element in X and call the result Z.
4. Multiply each element in Z and assign the result to a variable called Ω.
5. Compute $X^T Y - \Omega$.

Code
```
>> X = [3; 2; 6; 8]; Y = [4; 1; 3; 5];
% Question 1
>> Y + sum(X);
% Question 2
>> X.^Y;
% Question 3
>> Z = Y./X;
 % Question 4
```

```
>> Omega = Z.^2;
% Question 5
>> X'*Y-Omega;
```

3.3 Particular matrix

3.3.1 Diagonal matrix

M is said to be a diagonal matrix of size n, if and only if $m_{(i,j)} = 0 \; \forall i \neq j$, $(i,j) \in \{1 \cdots n\}^2$.

3.3.2 Upper triangular matrix (respectively, lower)

Assume that M is a square matrix. M is said to be upper triangular matrix (respectively, lower) if and only if $m_{(i,j)} = 0 \; \forall \, i < j$ (respectively, $i > j$).

3.3.3 Matrice diagonal (strictly) dominant

M is a square matrix of size n, if M verify the equation 3.12, then M is said (strictly) dominant.

$$\forall \, i \in \{1 \cdots n\} \sum_{j=1,\, j\neq i}^{n} |m_{ij}| \;\; \leq \;\; (\lneqq) \, |m_{ii}| \tag{3.12}$$

3.3.4 Hermitian matrix

M is a square matrix. M is hermitian if and only if $M^T = M^*$, where M^* denotes the conjugate matrix of M.

3.4 Linear systems

In automatic control, we encounter so many problems where there is a resolution of a generally linear system of n equations with n unknowns $x_1, \, x_2, \, \cdots, \, x_n$:

$$\begin{cases} a_{(1,1)}x_1 + a_{(1,2)}x_1 + \cdots + a_{(1,n)}x_1 & = b_1 \\ a_{(2,1)}x_1 + a_{(2,2)}x_1 + \cdots + a_{(2,n)}x_1 & = b_1 \\ \vdots & = \vdots \\ a_{(n,1)}x_1 + a_{(n,2)}x_1 + \cdots + a_{(n,n)}x_1 & = b_1 \end{cases} \tag{3.13}$$

This equation can be written in the matrix form, we have

$$\underbrace{\begin{pmatrix} a_{(1,1)} & a_{(1,2)} & \cdots & a_{(1,n)} \\ a_{(2,1)} & a_{(2,2)} & \cdots & a_{(2,n)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(n,1)} & a_{(n,2)} & \cdots & a_{(n,n)} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{b} \tag{3.14}$$

In the following, we are going to exhibit some three different methods for the resolution of a system having the same form as in equation 3.14.

3.4.1 Cramer method

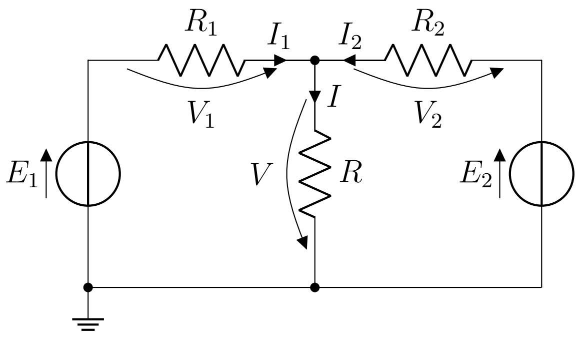


Figure 3.1: Basic electrical circuit

By applynig the Kirchhoff laws, we find the following system:

$$\begin{cases} E_1 - R_1 I_1 - RI &= 0 \\ E_2 - R_2 I_2 - RI &= 0 \\ I &= I_1 + I_2 \end{cases} \tag{3.15}$$

The previous system 3.15 can be written as follow:

$$\underbrace{\begin{pmatrix} E_1 \\ E_2 \\ 0 \end{pmatrix}}_{\text{b}} = \underbrace{\begin{pmatrix} R & R_1 & 0 \\ R & 0 & R_2 \\ 1 & -1 & -1 \end{pmatrix}}_{\text{A}} \underbrace{\begin{pmatrix} I \\ I_1 \\ I_2 \end{pmatrix}}_{\text{X}} \tag{3.16}$$

By applying the Cramer algorithm, the current I across R is:

$$I = \frac{\begin{vmatrix} E_1 & R_1 & 0 \\ E_2 & 0 & R_2 \\ 0 & -1 & -1 \end{vmatrix}}{|A|} \tag{3.17}$$

I is finally given by equation 3.18:

$$I = \frac{R_2E_1 + R_1E_2}{RR_1 + RR_2 + R_1R2} \tag{3.18}$$

The voltage V is given by equation 3.19:

$$V = \frac{RR_2E_1 + RR_1E_2}{RR_1 + RR_2 + R_1R2} \tag{3.19}$$

Remark 2

To compute the current I or the voltage V for the example pointed out in figure 3.1, application of Millman theorem would be easier. We just proceeded by Kirchoff laws in order to apply the Cramer method.

3.4.2 LU decomposition

This approach consists on writing a matrix as a product of two easy invertible matrices.

Given a system written in this form

$$AX = b. \tag{3.20}$$

The aim of the LU decomposition is to build the matrix A from a lower triangular matrix L and an upper triangular matrix such that U: $A = LU$. The elements on the diagonal of A have to be unities.

The procedure for decomposition is given as below

$$L = \begin{pmatrix} 1 & & & \\ l_{(2,1)} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{(n,1)} & l_{(n,2)} & \cdots & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_{(1,1)} & u_{(1,2)} & \cdots & u_{(1,n)} \\ & u_{(2,2)} & \cdots & u_{(2,n)} \\ & & \ddots & u_{(n-1,n)} \\ & & & u_{(n,n)} \end{pmatrix}. \tag{3.21}$$

For the resolution, the system 3.20 becomes:

$$\underbrace{A}_{LU} X = b \Leftrightarrow L \underbrace{UX}_{Y} = b \tag{3.22}$$

$$\Leftrightarrow \begin{cases} LY = b \\ UX = Y \end{cases} \tag{3.23}$$

3.4.3 Cholesky decomposition

This approach does not differ very much from the previous one since its aim is always to write the matrix A in equation 3.24 as the product of a lower triangular matrix L and an upper triangular matrix with the particular that the upper triangular matrix is L^T.

Given a system written in this form

$$AX = b. \tag{3.24}$$

The procedure for decomposition is given as below

$$L = \begin{pmatrix} l_{(1,1)} & & & \\ l_{(2,1)} & l_{(2,2)} & & \\ \vdots & \vdots & \ddots & \\ l_{(n,1)} & l_{(n,2)} & \cdots & l_{(n,n)} \end{pmatrix}, \tag{3.25}$$

For the resolution, the system 3.24 becomes:

$$\underbrace{A}_{LL^T} X = b \Leftrightarrow L \underbrace{L^T X}_{Y} = b \tag{3.26}$$

$$\Leftrightarrow \begin{cases} LY = b \\ L^T X = Y \end{cases} \tag{3.27}$$

3.5 In Closing

It results from what has been exposed in this chapter that the matrix representation seems to be the most adequate tool to represent some linear mappings between spaces (in our cases, it may be non linear). Such an investigation has to be motivated by a modest brief case of properties and tools in order to well conditioning your problem before resolving it.

CHAPTER 4

Statistics & Signal Processing

We must know, we will know.

David Hilbert

Contents

4.1 Statistics

We call random variable a variable which may have so many different values, without being able to determine with certitude their values.

Such variable is the consequence of a random experience.

a. Discrete random variable: We consider that the possible values of X are contained in a set of this form $\mathbb{X} = \{x_1, \cdots, x_N\}$. $\mathbb{X}$ has the finite cardinal N.

b. Continuous random variable: In the case of a continuous random variable, possibles values make now a compact continuous support. $\mathbb{X} \subset \mathbb{R}$

4.1.1 Expected value

The expected value, also called the first moment, is the term $\mathbb{E}[X]$, which is the sum of all possible values taken by the random variable X, weighted by their respective probability.

a. Discrete random variable:

$$\mathbb{E}[X] = \sum_{i \in \mathbb{X}} x_i P(X = x_i),$$

where $P(X = x_i)$ denotes the probability of the event $X = x_i$ to happen.

b. Continuous random variable:

$$\mathbb{E}[X] = \int_{\mathbb{X}} x f(x) dx,$$

where the function f denotes the density of probability.

The operator $\mathbb{E}[X]$ is a linear application from the set $\mathbb{X}$ to $\mathbb{R}$ ($\mathbb{E} : \mathbb{X} \to \mathbb{R}$).

Remark 1

If X is an $p \times l$ matrix, the expected value will be then a matrix of same size:

$$\mathbb{E}[X] = \mathbb{E}\left[\underbrace{\begin{pmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,l} \\ x_{2,1} & \cdots & x_{2,j} & \cdots & x_{2,l} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,l} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{p,1} & \cdots & x_{p,j} & \cdots & x_{p,l} \end{pmatrix}}_{\text{X}}\right] = \begin{pmatrix} \mathbb{E}[x_{1,1}] & \cdots & \mathbb{E}[x_{1,j}] & \cdots & \mathbb{E}[x_{1,l}] \\ \mathbb{E}[x_{2,1}] & \cdots & \mathbb{E}[x_{2,j}] & \cdots & \mathbb{E}[x_{2,l}] \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbb{E}[x_{i,1}] & \cdots & \mathbb{E}[x_{i,j}] & \cdots & \mathbb{E}[x_{i,l}] \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbb{E}[x_{p,1}] & \cdots & \mathbb{E}[x_{p,j}] & \cdots & \mathbb{E}[x_{p,l}] \end{pmatrix}$$

4.1.2 Variance

$$\begin{aligned} Var(X) &= Cov(X, X) \\ &= \mathbb{E}[(X-\mu_X)^2] \\ &= \mathbb{E}[X^2] - \mu_X^2 \end{aligned}$$

where μ_X denotes the expected value of X, $\mu_X = \mathbb{E}[X]$.

The variance of the random variable X is commonly referred as σ_X^2, where σ_X denotes the standard deviation of X.

4.1.3 Standard deviation

Assume that X is a random variable with mean value μ_X. The standard deviation of X is the square root of the variance of X. It can be expressed as follows:

$$\sigma_X = \sqrt{\mathbb{E}[(X-\mu_X)^2]} = \sqrt{Var(X)}$$

4.2 Signal processing

4.2.1 Convolution

The convolution is a mathematical operation between two signals. Let us for example consider two causals signals: $x(t)$ and $y(t)$.

$$x(t) * y(t) = y(t) * x(t) \tag{4.1}$$

$$= \int_0^t x(\tau)y(t-\tau)d\tau \tag{4.2}$$

The previous equation was written is the continuous domain. After discretization of this formula, we obtain the convolution definition formula in the discrete case. Hence, we will consider moments that are multiples of a period called the sample period, $t = nT$ where $n \in \mathbb{N}$.

$$x_n * y_n = y_n * x_n \tag{4.3}$$

$$= \sum_{i=0}^{n} x(i)y(n-i) \tag{4.4}$$

In Matlab, convolution is equivalent to a polynomial multiplication. For instance, consider the following polynomials[1]: $X(s) = s^2 + 2s + 3$ and $Y(s) = 4s^2 + 5s + 6$.

Code

```
>> X = [1 2 3]; Y = [4 5 6] % In descending power of s
>> Z = conv(X, Y) % Results are saved in the vector Z

Z =
4 13 28 27 18
```

4.2.2 Fourier transformer

The Fourier transform is a mathematical transform, very commonly used in physics and engineering applications. Its principle consists on transforming the mathematical function of time, $f(t)$ into a new function, denoted by $F(\nu)$. $F$is called the frequency spectrum of the function f. Given an input vector x of length N, the vector X, the DFT of x, is of same length and given by equation 4.5.

$$X(k) = \sum_{n=1}^{N} x(n)exp(\frac{-2j\pi(k-1)(n-1)}{N}, 1 \leq k \leq N \tag{4.5}$$

The inverse DFT is given as follows :

$$x(n) = \frac{1}{N}\sum_{k=1}^{N} X(k)exp(\frac{+2j\pi(k-1)(n-1)}{N}, 1 \leq n \leq N \tag{4.6}$$

[1]http://www.mathworks.com/help/matlab/math/convolution.html

CHAPTER 5

Scripts vs. Functions

Pure mathematics is, in its way,
the poetry of logical ideas.

Albert Einstein

Contents

5.1 Scripts

A script or a function in Matlab is an enumeration of Matlab instructions, saved in a file which has the extension ".m". That way, user can simply click on the run button ▶ or type the name of the file in order to let those commands evaluated on the command window[1].

Let consider the example of determining the reciprocal of a multidimensional function f, given as bellow:

$$\begin{array}{rccc} f: & \mathbb{C}^n & \longrightarrow & \mathbb{C}^n \\ & \vec{u} & \longmapsto & \vec{v} = f(\vec{u}) = M\vec{u} \end{array}$$

Our purpose is to calculate f^{-1} if it exists.

CODE

Code
```
% This script allows user to calculate
% the reciprocal of a reversible square function
M=input('Introduce the function f :\n');
N=size(M); % N=[Number of rows, Number of columns]
% Is f square?
if N(1)==N(2)
    if det(M)==0
        disp('F is not reversible');
    else % If yes, then test the reversibility of f
        disp('The reciprocal of f is :');
        disp(inv(M));
    end
else disp('Function must be square');
end
```

Last but not least, we have to test this script for different scenarios in order to be persuaded that the written code is valid in general case.

VALIDATION

Code
```
% First Test: f is reversible
Introduce the function f :
[1 2 4; 0.2 5 7*i+8; 4 -0.25 8]
The reciprocal of f is :
   0.2731 - 0.6204i  -0.0999 + 0.2553i   0.1867 + 0.1423i
   0.5991 - 0.2920i  -0.0470 + 0.1201i  -0.1474 + 0.0670i
  -0.1178 + 0.3011i   0.0485 - 0.1239i   0.0270 - 0.0691i
% Second Test: f is not reversible
Introduce the function f :
[1 2; 1 2]
F is not reversible
% Third Test: f is not square
Introduce the function f :
```

[1]For further reading, visit: http://www.mathworks.com/help/matlab/scripts.html

```
[0, 1, 5]
Function must be square
```

5.2 Functions

A function[2] [2] in Matlab is an m-file that starts with the key word *function*, followed by some output arguments, the function name, the input arguments and ends by *end*.

Compared to script, a function return results. It is loaded into RAM, that is why editing a function will not affect results until saving it[3].

We are going to treat in this section the example of determining the factorial of an integer number[4].

CODE

Code

```
function [ z ] = Fact(x)
% Fact(x) = x!, with x \in \mathbb{N}
if x>=0
    if x == 0
        z = 1;
    else z = x*Fact(x-1);
    end
else errordlg('Input must be an integer', 'Error');
end
end
```

VALIDATION

Code

```
>> Fact(3)
ans =
     6
```

Code

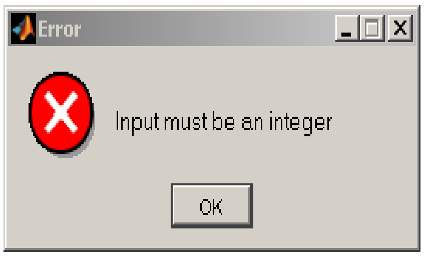
```
>> Fact(0)
ans =
     1
```

Let us test for instance, the factorial of a negative number, then an error dialog, figure 5.1, will be displayed.

[2]Take a look at: http://www.mathworks.com/help/matlab/functions.html
[3]Visit: http://www.isr.umd.edu/~austin/ence202.d/matlab-faq.html
[4]The factorial of $x \in \mathbb{N}$ is : $x! = x(x-1)\cdots 1 = x(x-1)!$, where $0! = 1$.

Code

```
>> Fact(-2)
```


Error

Input must be an integer

Figure 5.1: Error dialogue

Exercise 5

A Linear Feedback Shift Register (LFSR) is a shift register whose LSB is function (XOR in our case) of its previous state[a].
The main goal of this task is to develop in Matlab a script that generates a 4-bit Fibonacci LFSR, however, you may have to think for the portability of the script. (The feedback polynomial has to be easy to modify.)

[a] For more information, visit: http://en.wikipedia.org/wiki/Linear_feedback_shift_register

Code

```
% Take a look at <a href="matlab:
% web('http://en.wikipedia.org/wiki/Linear_feedback_shift_register')">
% The Wikipedia Web Site</a>.
P = [0 0 1 1]; % The feedback polynomial
N = length(P);
Y = ones(1, N); % Initialize the register
Reg = ones (1, N); % SEED
X = []; % An empty output vector
for i = 1:2^N-1
% Actual value of b is function of the previous LFSR's states
    b = rem(sum(Y.*P), 2);
    Reg = Y;
    for j = 2:N
        Y(j)=Reg(j-1); % Shift register
    end
    Y(1) = b;
    X = [X, Reg]; % The final output vector
```

```
end
```

The graph of the register output is as pointed out on figure 5.2. User can add

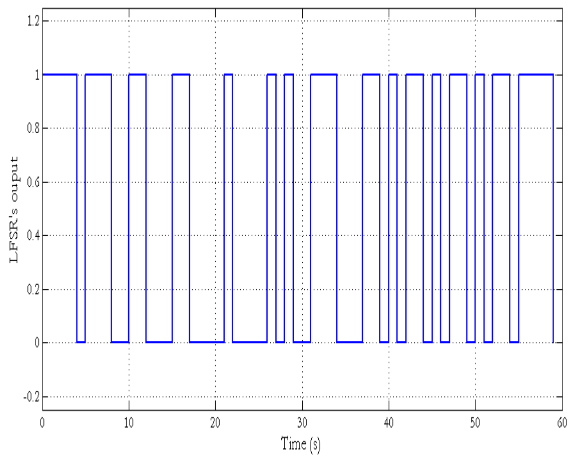


Figure 5.2: A 4-bit Fibonacci LFSR output graph

optional help to his code, just in the beginning of the file. For example, in this case, at the command prompt try help ScriptSBPA, you get this message: Take a look at The Wikipedia Web site.

Exercise 6

The Fibonacci numbers are computed according to equation 5.1:

$$F_n = F_{n-1} + F_{n-2}, \tag{5.1}$$

with $F_0 = F_1 = 1$.

1. Compute the first 10 Fibonacci numbers.
2. For the first 50 Fibonacci numbers, compute the ratio:

$$\frac{F_n}{F_{n-1}}. \tag{5.2}$$

3. Compare this ratio to the golden mean $\frac{1+\sqrt{5}}{2}$.

Code
```
function [F] = Fibo(n, X) % F_{n}(X)
% X is a vector
        if n == 0||n == 1
        F = 1;
        else
        F = Fibo(n-1, X)+Fibo(n-2, X);
        end
end
```

Code
```
>> for i = 1:10
Vect(i) = Fibo(i-1);
end
```

Code
```
>> for i = 1:50 % Do not try to test this code on your machine
Vect2(i) = Fibo(i)/ Fibo(i-1);
% Try tic, Fibo(i)/ Fibo(i-1); toc
end
```

Exercise 7

What does this code do ?

Code
```
Num = 0; Myeps = 1;
while (1+Myeps) > 1
        Myeps = Myeps/2;
        Num = Num + 1;
end
Num
Myeps = 2*Myeps
eps
```

The result shown in the Command Window is $2.2204e-016$. The main idea of this script is to count the number of multiple division of 1 by 2, which means:

$$Myeps = 2*(\frac{1}{2})^{Num}.$$

It is trivial that $Myeps$ is not negative, so normally $1+Myeps$ is always greater than zero. Thus, the condition used previously to trigger the while loop will always be true. The question is, so, how does Matlab had executed this code? To examine the answer, let us, first of all, made some definitions about the representation of number in Matlab, more precisely the floatting point in double precision[5]. Lecturer can take a see in [] and [] for more information. Any value stored in Matlab as double requires 64 bits, according to IEEE Standard 754, formatted as shown in the table 5.1. The command *realmin*, evaluated in

Table 5.1: Double precision formatting number

Bits	Significance
63	This bit represent the sign of the number (0 = positive, 1 negative)
62 $\rightarrow$ 52	signed 11 bits, range from $-1024 \rightarrow 1023$. They represent the exponent
51 $\rightarrow$ 0	Fraction of the number

the command window, gives the least number represented in double precision in Matlab:

Code
```
>> realmin
ans =

  2.2251e-308
```

The command *eps* represents the relative floating-point accuracy, it is the distance from 1.0 to the next larger double precision number[6].

Exercise 8

The Legendre polynomials, P_n, are defined by the following recurrence relation:

$$(n+1)P_{n+1}(x) - (2n+1)P_n(x) + nP_{n-1}(x) = 0, \tag{5.3}$$

with $P_0(x) = 1$, $P_1(x) = x$ and $P_2(x) = \frac{3x^2-1}{2}$. Compute the next three Legendre polynomials and plot all 6 over the interval $[-1, 1]$ on the same figure.

[5]See: http://www.mathworks.com/help/matlab/matlab_prog/floating-point-numbers.html

[6]Try » help eps

Code
```
function [P] = LegPol(n, X) % P_{n}(X)
     if n == 0
     P = ones(1, length(X));
     else if n == 1
           P = X;
           else
               if n == 2
               P = 1/2*(3*X.^2-1);
               else
               P = 1/n*((2*n-1)*LegPol(n-1, X)-(n-1)*LegPol(n-2, X));
               end
          end
     end
end
```

Code
```
>> X = -1:0.01:1; % Choose a step of 0.01 for more accuracy
>> for i = 1:6,
Vect(i, :) = LegPol(i-1, X);
end
```

The 6 graphics, corresponding to $P_i(X)$, $i = 0, \cdots, 5$, are shown on figure 5.3.

Code
```
>> plot(X, Vect(1, :), X, Vect(2, :), X, Vect(3, :),...
X, Vect(4, :), X, Vect(5, :), X, Vect(6, :));
>> legend('P_{0}(X)', 'P_{1}(X)', 'P_{2}(X)',...
'P_{3}(X)', 'P_{4}(X)', 'P_{5}(X)');
```

Exercise 9

Consider the following rational function:

$$y = \frac{1 - \frac{3}{5}x + \frac{3}{20}x^2 - \frac{1}{60}x^3}{1 + \frac{2}{5}x + \frac{1}{20}x^2}, \tag{5.4}$$

which approximates e^{-x} on the interval $x \in [0, 1]$. Implement this function for plotting and compare it with the exponential function.

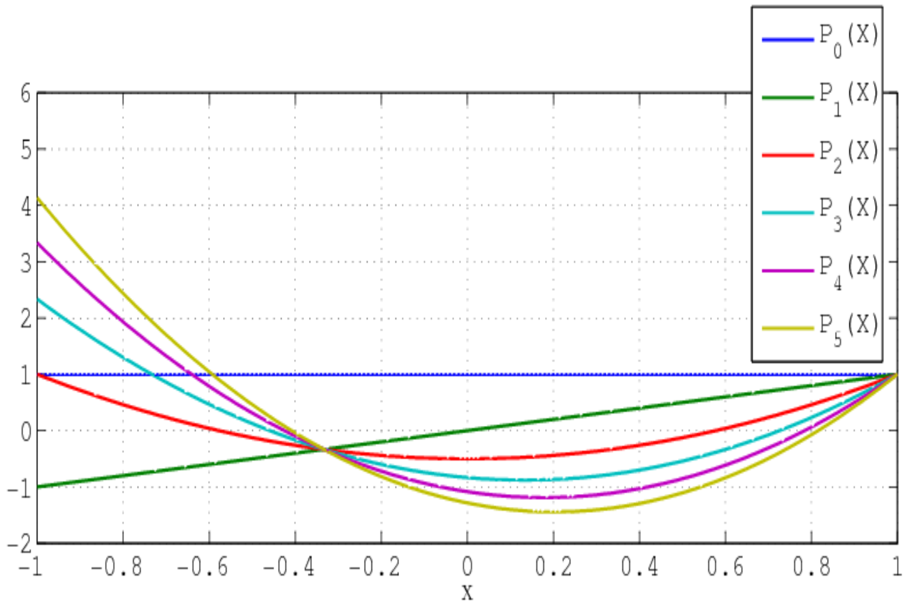


Figure 5.3: The Legendre polynomials solutions

Code
```
function [Y] = ExpApp(X)
Y = (1-3/5*X+3/20*X.^2-1/60*X.^3)./(1+2/5*X+1/20*X.^2)
end
```

Code
```
>> X = 0:0.1:1;
>> fplot(@(X)[ExpApp(X), exp(-X)], X)
```

CHAPTER 6

ODE: Ordinary Differential Equation

Mathematics is the queen of the sciences and number theory is the queen of mathematics.

Carl Friedrich Gauss (1777-1855)

Contents

In this chapter, we will study the resolution of Initial Value Problems (IVPs) using Matlab for ordinary differential equations[1]. Furthermore, lecturer can find more explanations on Boundary Value Problems (BVP) and Delay Differential Equations (DDE) in [3]. We assume throughout this chapter, because of

[1]See: http://www.mathworks.com/products/matlab/examples.html?file=/products/demos/shipping/matlab/odedemo.html

Matlab IVPs solvers[2], that the ODEs have the following form:

$$\dot{X}(t) = F(t, X), \tag{6.1}$$

where t is not necessarily a time parameter and differential of X with respect to t is denoted by $\dot{X}$. Similarly , we assume that the ODEs are defined on a finite interval: $t_0 \leqslant t \leqslant t_1$ and that initial conditions are given as a vector: $X(t_0) = X_0$. The first idea that comes to head is to compute approximations X_n on a mesh $t_0 < t_0 + h < \cdots < t_1$ where h is an infinitesimal displacement in parameter t. Basic methods are distinguished by whether or not they use previously computed quantities such as y_{n-k} where $k \in \{1 \cdots n\}$. If they do, they are named *methods with memory,* otherwise they are called *one-step methods* [3].

6.1 Theoretical reminder

Consider a first order system, described by this equation:

$$\tau\dot{y}(t) + y(t) = Ku(t) \tag{6.2}$$

The general solution of the previous system is the superposition of two solutions, one corresponding to homogeneous equation and the second to particular solution.

Homogeneous solution

the solution here show the influence of initial condition on the system without considering the input.

$$\tau\dot{y_h}(t) + y_h(t) = 0 \tag{6.3}$$

Integrating equation 6.3, we obtain

$$Ln(|y_h(t)|) = -\frac{t}{\tau} + A_0, \tag{6.4}$$

where Ln denotes the natural logarithm.
Applying the reciprocal function (Exponential function) to (6.4):

$$y_h(t) = A_1 \mathrm{e}^{-\frac{t}{\tau}}. \tag{6.5}$$

A_0 and A_1 are both real constant and $A_1 = \mathrm{e}^{A_0}$.
Considering the initial condition $y(t = t_0) = y_0$, the constant A_1 is equal to $y_0 \mathrm{e}^{\frac{t_0}{\tau}}$. Then, the homogenous solution becomes:

$$y_h(t) = y_0 \mathrm{e}^{-\frac{t-t_0}{\tau}} \tag{6.6}$$

[2]Take a look at: http://pundit.pratt.duke.edu/wiki/MATLAB:Ordinary_Differential_Equations/Examples

Particular solution

It corresponds to the non-homogeneous equation, given by equation 6.2. For this reason, we consider the equation 6.6 and using the method of variation of parameters, we obtain a particular solution of the form:

$$y_p(t) = A(t)\text{e}^{-\frac{t}{\tau}} \tag{6.7}$$

Differentiating $y_p(t)$ with respect to time t gives:

$$\dot{y}_p(t) = \dot{A}(t)\text{e}^{-\frac{t}{\tau}} + A(t)\frac{\partial \text{e}^{-\frac{t}{\tau}}}{\partial t} \tag{6.8}$$

After development, $\dot{y}_p(t)$ becomes as follows:

$$\dot{y}_p(t) = \dot{A}(t)\text{e}^{-\frac{t}{\tau}} - \frac{1}{\tau}A(t)\text{e}^{-\frac{t}{\tau}} \tag{6.9}$$

Replacing $y_p(t)$ in equation 6.2 by its equivalent given in equation 6.9 give:

$$\tau(\dot{A}(t) - \frac{1}{\tau}A(t))\text{e}^{-\frac{t}{\tau}} + A(t)\text{e}^{-\frac{t}{\tau}} = Ku(t) \tag{6.10}$$

After simplification, we obtain:

$$\tau\dot{A}(t)\text{e}^{-\frac{t}{\tau}} = Ku(t) \tag{6.11}$$

From the previous equation, we deduce the expression of $\dot{A}(t)$:

$$\dot{A}(t) = \frac{K}{\tau}\text{e}^{\frac{t}{\tau}}u(t) \tag{6.12}$$

After integration above the interval $[t_0\, t]$,

$$A(t) = \int_{t_0}^{t} \frac{K}{\tau}\text{e}^{\frac{\mu}{\tau}}u(\mu)d\mu \tag{6.13}$$

By substituting (6.13) in (6.14):

$$y_p(t) = \int_{t_0}^{t} \frac{K}{\tau}\text{e}^{-\frac{t-\mu}{\tau}}u(\mu)d\mu \tag{6.14}$$

Ultimately, the global solution $y = y_p + y_h$ is as follows:

$$y(t) = y_0\text{e}^{-\frac{t-t_0}{\tau}} + \int_{t_0}^{t} \frac{K}{\tau}\text{e}^{-\frac{t-\mu}{\tau}}u(\mu)d\mu \tag{6.15}$$

Let us consider a linear system, with order greater than one, for example n: The system equation can be written for instance on this form:

$$\sum_{i=0}^{n} a_i y^{(i)}(t) = \sum_{j=0}^{m} b_j u^{(j)}(t) \tag{6.16}$$

We assume that $a_n = 1$ and $m \leqslant n$ otherwise system is an anticipatory action, which is not our physical studies cases. Equation 6.16 can be then transformed on the following form:

$$y^{(n)}(t) = -\sum_{i=0}^{n-1} a_i y^{(i)}(t) + \sum_{j=0}^{m} b_j u^{(j)}(t) \tag{6.17}$$

The above system can be written in the matrix form as follows:

$$\begin{cases} \dot{X}(t) &= AX(t) + Bu(t) \\ y(t) &= CX(t) + Du(t) \end{cases} \tag{6.18}$$

$(A,\, B,\, C,\, D) \in \mathbb{M}_{(n,\,n)}(\mathbb{C})x\mathbb{M}_{(n,\,1)}(\mathbb{C})x\mathbb{M}_{(1,\,n)}(\mathbb{C})x\mathbb{M}_{(1,\,1)}(\mathbb{C})$

D is a matrix input-output direct transfer, it exists if and only if $m = n$, otherwise, this term is zero.

$X(t)$ here denotes the variable states of the system. It is a vector of n elements which presents the minimum memory capacity that system can save in order to be able to determine its further evolution. It may have or not a physical significance.

C and B are respectively the output and the input vectors. It represents the influence of inputs on states, respectively, of states on outputs.

A is the state matrix, it connects the states vector $X(t)$ to its velocity $\dot{X}(t)$. The figure 6.1 shows a state-space description scheme given in [4]: For further

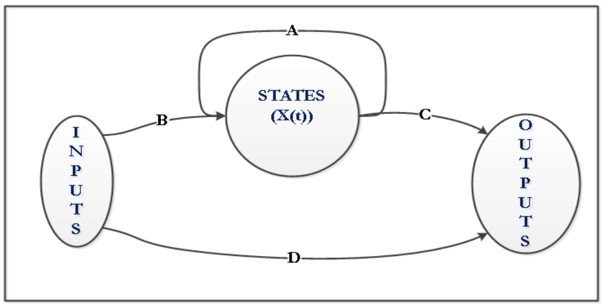


Figure 6.1: State Space Representation [4]

reading, lecturer can wait for the upcoming course entitled ”Mutivariable System Theory and Design” or simply consult [4]. We consider the first form of the canonical form. Given the solution of equation 6.15, states $X(t)$ can be determined as follows:

$$X(t) = \mathrm{e}^{A(t-t_0)} X_0 + \int_{t_0}^{t} \mathrm{e}^{A(t-\mu)} Bu(\mu) d\mu \tag{6.19}$$

The idea behind this recap is to show that is possible to transform a differential equation of order n to n equations of first order, which means, in the matrix representation a first order system. From the equation 6.19, we observe that quantity $e^{-A(t_0-\mu)}Bu(\mu)$ is integrable if and only if the matrix $\xi = [B\ AB\ ..\ A^{n-1}B]$ has a full rank, otherwise, it does not exist any input u that takes states $X(t)$ from arbitrary initial conditions to desired values in a finite time. This means that the system is not fully controllable, depending on the rank value of ξ.

Let us assume that m in equation 6.20 is strictly inferior then n, which means that matrix D in system 6.18 is zero.

$$y^{(n)}(t) = -\sum_{i=0}^{n-1} a_i y^{(i)}(t) + \sum_{j=0}^{m} b_j u^{(j)}(t) \tag{6.20}$$

6.2 Numerical methods

6.2.1 Numerical approximation

a) Euler methods

1. Forward Euler
   Let consider the following equation, given by:

$$\dot{y} = f(t, y), \tag{6.21}$$

with an initial condition $y_0 = t_0$. The rate $\dot{y}$ can be approximated by:

$$\dot{y}_n \approx \frac{y_{n+1} - y_n}{h}. \tag{6.22}$$

Then, equation 6.21 becomes:

$$y_{n+1} = y_n + hf(y_n, t_n), \tag{6.23}$$

where $t_n \in \{t_0 ... t_1\}$ with a step equals to h.

2. Backward Euler
   The same demonstration can be followed, just change the forward difference approximation this time by a backward one, hence, $\dot{y}_n$ becomes:

$$\dot{y}_n \approx \frac{y_n - y_{n-1}}{h}. \tag{6.24}$$

$$y_n = y_{n-1} + hf(y_n, t_n) \tag{6.25}$$

   This method is as accurate as the forward Euler method [?].

3. Modified Euler
   One third method consists on applying the trapezoidal rule to the solution of equation 6.21.

$$y_{n+1} = y_n + \frac{h}{2}(f(y_{n+1}, t_{n+1}) + f(y_n, t_n)) \tag{6.26}$$

   This method is more stable and accurate than the previous approach [?].

6.2.2 Built-in functions

a) Runge-Kutta methods

Given the equation 6.27, where y is an unknown function:

$$\dot{y} = f(t, y), \tag{6.27}$$

with an initial condition $y_0 = t_0$. The equation 6.27 shows that the rate at which y changes, referred as $\dot{y}$, is a function of t and y itself. The most well-known method used, often called the **classical Runge-Kutta method** or simply **RK4**, is as follows, we compute the terms recursively as [?]:

$$\begin{cases} k_1 = hf(t_n, y_n) \\ k_2 = hf(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 = hf(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 = hf(t_n + h, y_n + k_3) \end{cases}$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{6.28}$$

where $t_n \in \{t_0 ... t_1\}$ with a step equals to h.

Remark 1

The RK4 method is a fourth-order method, meaning that the error per step is on the order of $\mathcal{O}(h^5)$, while the total accumulated error has order $\mathcal{O}(h^4)$.[a]

[a]Visit: en.wikipedia.org/wiki/Runge-Kutta_methods

b) Using ODE45

Matlab involves many types of solvers when resolving ODE. We mention here the *ode45* used most of the time and should be the first to try.

SYNOPSIS

```
[t,x] = ode45('xprime',t0,tf,x0,tol,trace);
```

EXAMPLE N°1

Consider this differential equation, given by:

$$\dot{y}(t) = \frac{cos(t)}{2y - 2}. \tag{6.29}$$

This function should be written in an m-file in Matlab.

Code
```
function Doty = MyFirstODE(t, y)
% dy/dt = cos(t)/(2y-2)
     Doty = cos(t)/(2*y-2);
end
```

At the command window, strike the following instruction in order to resolve the equation 6.29.

Code
```
>> [T, X] = ode45(@MyFirstODE, [0 6.3], 3);
```

To observe the trajectory of the scalar y, think of the command *plot*.

Code
```
>> plot(T, X, 'LineWidth',2); grid on;
```

In order to insert title to the current figure and add labels, respectively for the x-axis and y-axis, these commands have to be evaluated.

Code
```
>> title('X versus T', 'FontName', 'Garamond', ...
'FontSize', 14, 'FontWeight', 'Bold');
>> xlabel('T (s)', 'FontName', 'Garamond', ...
'FontSize', 14, 'FontWeight', 'Bold');
>> ylabel('X', 'FontName', 'Garamond', ...
'FontSize', 14, 'FontWeight', 'Bold');
```

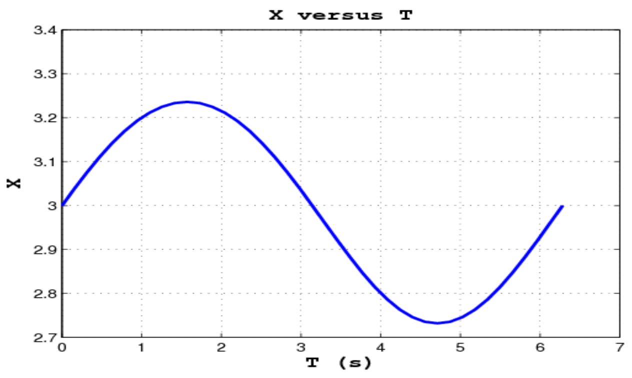


Figure 6.2: Trajectory of equation 6.29's solution

EXAMPLE N°2

Consider the following matrix representation:

$$\dot{X}(t) = \begin{bmatrix} 0 & 1 \\ -3 & -0.5 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{6.30}$$

Obviously, this is a second order system but for Matlab, nothing has to be changed compared to the previous file. Below, it is the function declaration:

Code
```
function [DotX] = My2ndODE( t, X )
% dX(t)/dt = A*X(t)+B*u(t)
     A = [0 1; -3 -0.5]; B = [0; 1]; u = exp(-t);
     DotX = A*X+B*u;
end
```

At the Matlab command prompt, if user try the following command, he will get the graphic indicated by figure 6.3.

Code
```
>> [T, X] = ode45(@My2ndODE, [0 5], [0.3; 0]);
```

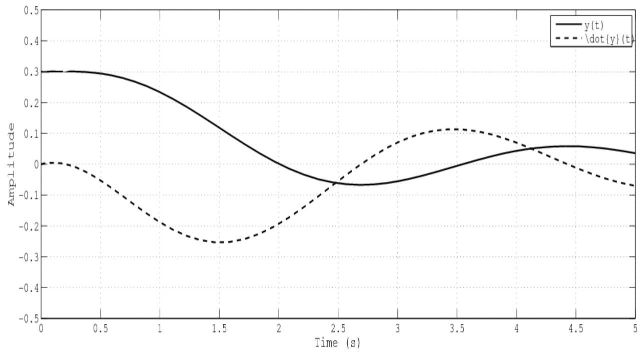


Figure 6.3: A second order differential equation resolution

CHAPTER 7

Multivariable System Theory and Design

We are servants rather than
masters in mathematics.

Charles Hermite (1822-1901)

Contents

TRANSFER FUNCTION allows only to show the relationship between the input and the output for a given system. It does not take into consideration the evolutions of some intrinsic variables, called states. The state space description is more convenient to represent a physical system than the transfer function.

We consider, hereafter, the case of a linear Single Input-Single Output system (If not mentioned otherwise). This system is described by the following differential equation, where $m \leq n$:

$$\sum_{i=0}^{n} a_i \frac{d^i y(t)}{dt^i} = \sum_{j=0}^{m} b_j \frac{d^j u(t)}{dt^j}. \tag{7.1}$$

Applying the Laplace transformer to the previous equation leads to the transfer function written in the operational domain, given as follow, where: $m \leq n$:

$$\frac{Y(s)}{U(s)} = \frac{\sum_{j=0}^{m} b_j s^j}{\sum_{i=0}^{n} a_i s^i}. \tag{7.2}$$

7.1 State space representation

7.1.1 Canonical forms

a) Compagnion form

FIRST FORM

We already mentioned in chapter 6 that state vector for a given system presents the minimum memory capacity that can save in order to be able to determine its further evolution. That is why, in the transfer function, we will proceed to show the integrative form $\frac{1}{s}$, not the derivative one s. Hence, given the equation 7.3:

$$\frac{Y(s)}{U(s)} = \frac{\sum_{j=0}^{m} b_j s^j}{\sum_{i=0}^{n} a_i s^i} = \frac{\sum_{j=0}^{m} b_j (\frac{1}{s})^{n-j}}{\sum_{i=0}^{n} a_i (\frac{1}{s})^{n-i}}, \tag{7.3}$$

we set $a_n = 1$ and multiply it by the term $\frac{1}{s^n}$ in order to avoid any derivative scheme in the realization of the block diagram equivalent to the considered system. Result is given by equation 7.4.

$$\frac{Y(s)}{U(s)} = \frac{\sum_{j=0}^{m} b_j (\frac{1}{s})^{n-j}}{1 + \sum_{i=0}^{n-1} a_i (\frac{1}{s})^{n-i}} \tag{7.4}$$

After linearizing the equation 7.4, we obtain:

$$Y(s) + \sum_{i=0}^{n-1} a_i (\frac{1}{s})^{n-i} Y(s) = \sum_{j=0}^{m} b_j (\frac{1}{s})^{n-j} U(s) \tag{7.5}$$

The output $Y(s)$ is accessible and given by equation 7.6.

$$Y(s) = \sum_{j=0}^{m} b_j(\frac{1}{s})^{n-j}U(s) - \sum_{i=0}^{n-1} a_i(\frac{1}{s})^{n-i}Y(s) \tag{7.6}$$

$$= \sum_{j=0}^{m} b_j(\frac{1}{s})^{n-j} \underbrace{[U(s) - \frac{\sum_{i=0}^{n-1} a_i(\frac{1}{s})^{n-i}}{\sum_{j=0}^{m} b_j(\frac{1}{s})^{n-j}}Y(s)]}_{W(s)} \tag{7.7}$$

Let us intercalate a new variable $W(s)$ as:

$$\begin{aligned} W(s) &= U(s) - \frac{\sum_{i=0}^{n-1} a_i(\frac{1}{s})^{n-i}}{\sum_{j=0}^{m} b_j(\frac{1}{s})^{n-j}}Y(s) \\ &= U(s) - \sum_{i=0}^{n-1} a_i(\frac{1}{s})^{n-i}\frac{Y(s)}{\sum_{j=0}^{m} b_j(\frac{1}{s})^{n-j}} \\ &= U(s) - \sum_{i=0}^{n-1} a_i(\frac{1}{s})^{n-i}W(s) \\ &= U(s) - \sum_{i=0}^{n-1} a_i(\frac{1}{s})^{n-i}W(s) \end{aligned} \tag{7.8}$$

The output $Y(s)$ is then given by equation 7.9.

$$Y(s) = \sum_{j=0}^{m} b_j \underbrace{(\frac{1}{s})^{n-j}W(s)}_{x_k(s)} \tag{7.9}$$

$$= \sum_{k=0}^{m} b_k x_k(s) \tag{7.10}$$

The state matrix, input and output vectors are given in equations 7.11 and 7.12.

$$A_c = \begin{pmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 0 & 1 \\ -a_0 & -a_1 & \cdots & \cdots & -a_{n-2} & -a_{n-1} \end{pmatrix}, \qquad B_c = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \tag{7.11}$$

$$C_c = \begin{pmatrix} b_0 & \cdots & b_m & 0 & \cdots & 0 \end{pmatrix}. \tag{7.12}$$

Consider for example that we have a system written as follow:

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{cases}$$

In order to be able to write the previous description in the same form as in equation **??**, that means the canonical form, it is possible to determine a base changing $z_c(t) = Px(t)$ where $z_c(t)$ is the state vector with respect to the newest base.

$$\begin{cases} \dot{z}_c(t) &= A_c z_c(t) + B_c u(t) \\ y(t) &= C_c z_c(t) \end{cases}$$

where:

$$A_c = PAP^{-1}, \qquad B_c = PB, \qquad C_c = CP^{-1},$$

and the matrix P is calculated as bellow:

$$P = \begin{pmatrix} P_1 \\ P_1 A \\ \vdots \\ P_1 A^{n-1} \end{pmatrix}, \quad P_1 = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} B & AB & \cdots & A^{n-1}B \end{pmatrix}^{-1}.$$

Remark 1

This form is baptized also as the canonical form for controllability. It is commonly used for continuous system.

SECOND FORM

$$Y(s) = \sum_{j=0}^{m} b_j(\frac{1}{s})^{n-j} U(s) - \sum_{i=0}^{n-1} a_i(\frac{1}{s})^{n-i} Y(s) \tag{7.13}$$

The state vector X can be defined as follow:

$$X(s) = \begin{pmatrix} \qquad\qquad & Y(s) \end{pmatrix}$$

The state matrix, input and output vectors are given in equations 7.14 and 7.15.

$$A_o = \begin{pmatrix} 0 & 0 & 0 & \cdots & \cdots & -a_0 \\ 1 & \ddots & 1 & \ddots & \ddots & -a_1 \\ 0 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 & 0 & -a_{n-2} \\ 0 & \cdots & \cdots & \cdots & 1 & -a_{n-1} \end{pmatrix}, \qquad B_o = \begin{pmatrix} b_0 \\ \vdots \\ b_m \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{7.14}$$

$$C_o = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix}. \tag{7.15}$$

Remark 2

This form is baptized also as the canonical form for observability, referred also as Frobenius form. It is commonly used for discret system.

b) Modal form

a. Simple poles

$$\begin{aligned} H(s) = \frac{Y(s)}{U(s)} &= \frac{\sum_{j=0}^{m} b_j s^j}{\sum_{i=0}^{n} a_i s^i}, m \leq n \\ &= \sum_{k=1}^{n} \frac{\alpha_k}{s - s_k} \end{aligned} \tag{7.16}$$

$$A = \begin{pmatrix} s_1 & 0 & \cdots & \cdots & 0 \\ 0 & s_2 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & s_n \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \cdots & \alpha_n \end{pmatrix}. \tag{7.17}$$

- Example

Consider the state space representation in equation 7.18:

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3.5 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} 2 & 0.1 & -1 & 2 \end{pmatrix}. \tag{7.18}$$

The corresponding transfer function is :

$$H(s) = \frac{2}{s+1} + \frac{0.1}{s+2} + \frac{-1}{s-1} + \frac{2}{s-3.5} \tag{7.19}$$

b. Multiple poles

$$\begin{aligned} H(s) = \frac{Y(s)}{U(s)} &= \frac{\sum_{j=0}^{m} b_j s^j}{\sum_{i=0}^{n} a_i s^i}, m \leq n \\ &= \frac{\alpha_1}{(s - s_1)^2} + \frac{\alpha_2}{s - s_1} + \sum_{k=2}^{n-1} \frac{\alpha_k}{s - s_k} \end{aligned} \tag{7.20}$$

Assume that s_1 is a double pole. Then the corresponding state representation becomes as follow:

$$A = \begin{pmatrix} s_1 & 0 & \cdots & \cdots & 0 \\ 0 & s_1 & 0 & \cdots & 0 \\ \vdots & \ddots & s_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & s_{n-1} \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ \vdots \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \cdots & \alpha_n \end{pmatrix}. \tag{7.21}$$

- Example

Consider the state space representation in equation 7.22:

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad C = \begin{pmatrix} 2 & 0.1 & -3.5 & 5 & 1 \end{pmatrix}. \tag{7.22}$$

The corresponding transfer function is :

$$H(s) = \frac{2}{s+1} + \frac{0.1}{(s+1)^2} + \frac{-3.5}{(s+1)^3} + \frac{5}{s-3} + \frac{1}{(s-3)^2} \tag{7.23}$$

7.1.2 Passage from state equation to matrix transfer function

Given a system described by the equation 7.24.

$$\begin{cases} \dot{X}(t) &= AX(t) + Bu(t) \\ y(t) &= CX(t) + Du(t) \end{cases} \tag{7.24}$$

The matrix transfer function is obtained as follows, where $Y(s)$ and $U(s)$ denote respectively the Laplace transformers of $y(t)$ and $u(t)$:

$$\frac{Y(s)}{U(s)} = C(sI_n - A)^{-1}B + D, \tag{7.25}$$

The matrix transfer function is unique. It has as many rows as the number of outputs and as many columns as the number of inputs.

Exercise 10

Consider the following representation corresponding to a MIMO system:

$$\begin{cases} \dot{X}(t) & = AX(t) + Bu(t), \\ y(t) & = CX(t) + Du(t), \end{cases} \tag{7.26}$$

where:

$$A = \begin{pmatrix} 2 & 1 \\ -4 & 0.35 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 \\ 0 & 3.2 \\ 4 & 0.5 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 \\ 0.15 & 1 \\ 2 & 0.25 \end{pmatrix}.$$

Calculate the matrix transfer function.

Assume that:

$$B = \begin{array}{c} \begin{pmatrix} \boxed{\begin{matrix} 1 \\ 0 \end{matrix}} & \boxed{\begin{matrix} 1 \\ 1 \end{matrix}} \end{pmatrix} \\ \begin{matrix} b_1 \uparrow & \uparrow b_2 \end{matrix} \end{array} \qquad C = \begin{array}{cl} \begin{pmatrix} \boxed{1 \quad 1} \\ \boxed{0 \quad 3.2} \\ \boxed{4 \quad 0.5} \end{pmatrix} & \begin{matrix} \leftarrow c_1 \\ \leftarrow c_2 \\ \leftarrow c_3 \end{matrix} \end{array}$$

Since the dimension of the transfer matrix D is $(3, 2)$, then the system 7.26 has 3 outputs and 2 inputs. Let us call $u(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$ and $y(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix}$.

$$\begin{pmatrix} Y_1(s) \\ Y_2(s) \\ Y_3(s) \end{pmatrix} = \begin{pmatrix} \dfrac{Y_1(s)}{U_1(s)} & \dfrac{Y_1(s)}{U_2(s)} \\ \\ \dfrac{Y_2(s)}{U_1(s)} & \dfrac{Y_2(s)}{U_2(s)} \\ \\ \dfrac{Y_3(s)}{U_1(s)} & \dfrac{Y_3(s)}{U_2(s)} \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} c_1(sI_2 - A)^{-1}b_1 + D(1,1) & c_1(sI_2 - A)^{-1}b_2 + D(1,2) \\ \\ c_2(sI_2 - A)^{-1}b_1 + D(2,1) & c_2(sI_2 - A)^{-1}b_2 + D(2,2) \\ \\ c_3(sI_2 - A)^{-1}b_1 + D(3,1) & c_3(sI_2 - A)^{-1}b_2 + D(3,3) \end{pmatrix}}_{\text{Matrix Transfer Function: } M} \begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix}$$

Exercise 11

$$\begin{cases} \dot{X}(t) &= \begin{pmatrix} 0 & 1 \\ -5 & -4 \end{pmatrix} X(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) \\ \\ y(t) &= \begin{pmatrix} 3 & 4 \end{pmatrix} X(t) \end{cases}$$

1. Without a prior computing, determine the transfer function. Justify?
2. Calculate the step response of the system for a zero initial condition.
3. Compute the state vector $X(t)$ generated from a zero input and initial conditions $X(0^+) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

1. Since the previous system is written is the canonical form, the transfer function is given by the equation 7.27:

$$\begin{aligned} H(s) &= \frac{Y(s)}{U(s)} \\ &= \frac{3s+4}{s^2+4s+5} \end{aligned} \tag{7.27}$$

2. The output $Y(s)$ is given by:

$$Y(s) = H(s)U(s) \tag{7.28}$$
$$= \frac{3s+4}{s(s^2+4s+5)} \tag{7.29}$$
$$= \frac{\alpha}{s} + \frac{\beta}{s-s_1} + \frac{\delta}{s-s_2} \tag{7.30}$$

where:

$$s_1 = -2 + \jmath \qquad s_2 = -2 - \jmath = s_1^*$$
$$s_1 + s_2 = -4 \qquad s_1 - s_2 = 2\jmath \qquad s_1 s_2 = 5$$

$$\begin{aligned} \alpha &= s\frac{3s+4}{s(s^2+4s+5)}_{|s=0} &= \frac{3}{5} \\ \beta &= (s-s_1)\frac{3s+4}{s(s^2+4s+5)}_{|s=s_1} &= \frac{3+4s_1}{s_1(s_1-s_2)} \\ \delta &= (s-s_2)\frac{3s+4}{s(s^2+4s+5)}_{|s=s_2} &= \frac{3+4s_2}{s_2(s_2-s_1)} \end{aligned}$$

$$y(t) = (\alpha + \underbrace{\beta \mathsf{e}^{s_1 t} + \delta \mathsf{e}^{s_2 t}}_{r(t)})\Gamma(t) \tag{7.31}$$

$$\begin{aligned}
r(t) &= \frac{1}{s_1 s_2 (s_1 - s_2)}(s_2(3+4s_1)\mathsf{e}^{s_1 t} - s_1(3+4s_2)\mathsf{e}^{s_2 t}) && (7.32)\\
&= \frac{\mathsf{e}^{-2t}}{10\jmath}((-2-\jmath)(-5+4\jmath)\mathsf{e}^{\jmath t} - (-2+\jmath)(-5+4\jmath)\mathsf{e}^{-\jmath t}) && (7.33)\\
&= \frac{\mathsf{e}^{-2t}}{10\jmath}(14(\mathsf{e}^{\jmath t} - \mathsf{e}^{-\jmath t} - 3\jmath(\mathsf{e}^{\jmath t} + \mathsf{e}^{-\jmath t})) && (7.34)\\
&= \frac{\mathsf{e}^{-2t}}{5}(14\,\mathsf{sin}(t) - 3\,\mathsf{cos}(t)) && (7.35)
\end{aligned}$$

$$y(t) = [\frac{3}{5} + \frac{\mathsf{e}^{-2t}}{5}(14\,\mathsf{sin}(t) - 3\,\mathsf{cos}(t))]\Gamma(t) \tag{7.36}$$

3. Since $X(0^+) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, the system description becomes as follow:

$$\dot{X}(t) = \underbrace{\begin{pmatrix} 0 & 1 \\ -5 & -4 \end{pmatrix}}_{A} X(t) \tag{7.37}$$

$$X(t) = \mathsf{e}^{At} X(0^+) \tag{7.38}$$

Now, let us compute the exponential matrix e^{At}. The matrix A can be written as PDP^{-1}, where D is its diagonal form. Hence $D = \begin{pmatrix} -2+\jmath & 0 \\ 0 & -2-\jmath \end{pmatrix}$ and $P = \begin{pmatrix} -0.3651 - 0.1826\jmath & -0.3651 + 0.1826\jmath \\ 0.9129 & 0.9129 \end{pmatrix}$. The inverse of the matrix P is then equal to $\begin{pmatrix} 2.7386\jmath & 0.5477 + 1.0954\jmath \\ -2.7386\jmath & 0.5477 - 1.0954\jmath \end{pmatrix}$. The term e^{At} is calculated as follow:

$$\begin{aligned}
X(t) &= P\mathsf{e}^{Dt}P^{-1}X(0^+) && (7.39)\\
& && (7.40)\\
&= \begin{pmatrix} \mathsf{e}^{t(-2-\jmath)}\left(\frac{1}{2} + 2\jmath\right) + \mathsf{e}^{t(-2+\jmath)}\left(\frac{1}{2} - 2\jmath\right) \\ \mathsf{e}^{t(-2-\jmath)}\left(1 - \frac{9}{2}\jmath\right) + \mathsf{e}^{t(-2+\jmath)}\left(1 + \frac{9}{2}\jmath\right) \end{pmatrix}\Gamma(t) && (7.41)\\
&= \begin{pmatrix} \mathsf{e}^{-2t}(\mathsf{cos}\,(t) + 4\,\mathsf{sin}\,(t)) \\ \mathsf{e}^{-2t}(2\,\mathsf{cos}\,(t) - 9\,\mathsf{sin}\,(t)) \end{pmatrix}\Gamma(t) && (7.42)
\end{aligned}$$

Remark 3

For a system of order n, it does exist at least n^2 possible state description.

7.2 Controllability and Observability

7.2.1 Controllability

Before designing control laws for linear systems, a question must be asked:

Does exist a control law that satisfies the desire of the customer ?

To answer this, a study of the plant controllability has to be done.

a) Point-wise state controllability (p.s)

Let us recall the description of the vector of states $X(t)$ for a particular t:

$$X(t) = \mathrm{e}^{A(t-t_0)}X_0 + \int_{t_0}^{t} \mathrm{e}^{A(t-\mu)}Bu(\mu)d\mu \tag{7.43}$$

Setting $X(t) = 0$, then for some $t > t_0$, we obtain:

$$X_0 = -\int_{t_0}^{t} \mathrm{e}^{-A\mu}Bu(\mu)d\mu \tag{7.44}$$

The equation 7.44 can be solved for $u(\mu)$ for any given X_0 if, and only if, the rows of the matrix $\mathrm{e}^{-A\mu}B$ are linearly independent. Given the Taylor approximation, the matrix exponential can be expressed in terms of $I_n, A, A^2, \cdots, A^k, \cdots, A^\infty$. Since every matrix verifies its characteristic polynomial, the term e^{A} can be written in terms of $I_n, A, A^2, \cdots, A^{n-1}$.

A necessary and sufficient condition for point- wise state controllability is that the matrix ξ has a full rank, where ξ is the matrix:

$$\xi = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix} \tag{7.45}$$

Exercise 12

A factor in determining useful life of a flexible structure, such as ship, a tall building, or a large airplane, is the possibility of fatigue failures due to structural vibrations. Each vibration mode is described by an equation of the form $m\ddot{x}+kx = u(t)$, where $u(t)$ is the input force. Is it possible to find an input which will drive both the deflection $x(t)$ and the velocity $\dot{x}(t)$ to zero in finite time for arbitrary initial conditions?

We assume that $y_1(t) = x(t)$ and $y_2(t) = \dot{x}(t)$. Hence, the system description becomes as follows:

$$\left\{ \begin{bmatrix} \dot{y}_1(t) \\ \dot{y}_2(t) \end{bmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{pmatrix}}_{\text{A}} \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix}}_{\text{B}} u(t) \right.$$

What you have to do now, is to compute the rank of the matrix $[B, AB]$.

$$[B, AB] = \begin{pmatrix} 0 & \frac{1}{m} \\ \frac{1}{m} & 0 \end{pmatrix} \tag{7.46}$$

The previous matrix, computed in equation 7.46 is not singular, hence, the system described by the equation $m\ddot{x} + kx = u(t)$ is fully controllable, which means it exists a command $u(t)$ that drives both the deflection $x(t)$ and the velocity $\dot{x}(t)$ to zero in finite time from arbitrary initial conditions.

b) Functional controllability (f)

A system with l outputs and m inputs is functional controllable, or controllable (f), if there exists a sequence of inputs defined for $t > t_0$, wich generates a suitable vector of outputs from the initial conditions X_0.

a. If $l > m$ then the system is not functionally controllable

b. If $l = m$ then a necessary and sufficient condition for functional controllabilty is that its transfer-function matrix is not singular.

c. If $l < m$ then the system is functionally controllable if, and only if there exists at least one non-zero $l \times l$ minor of its transfer-function matrix is not singular.

Functional controllability does not imply pointwise-state controllability and vice versa.

7.2.2 Observability

A system is said to be observable if there exists a time $t_1 > t_0$ such that given the input $u(t)$ and the ouput $y(t)$, it is possible to deduce the initial conditions X_0. A necessary and sufficient condition for observability is that the the matrix O has a full rank, where O is the matrix:

$$O = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{7.47}$$

7.3 State Control Feedback Law

We consider the system described by equation 7.48.

$$\begin{cases} \dot{X}(t) &= AX(t) + Bu(t) \\ y(t) &= CX(t) \end{cases} \tag{7.48}$$

The equivalent bloc diagram is given by figure 7.1. The command law consists

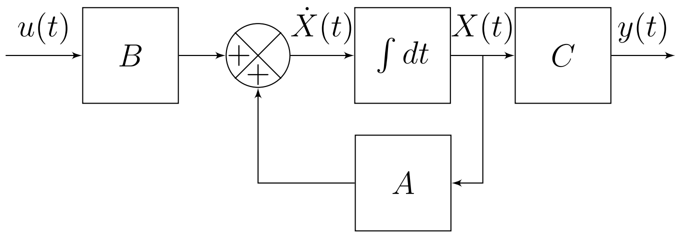


Figure 7.1: Bloc diagram of the system

on applying the value of the vector state $X(t)$ in the command definition $u(t)$. Hence, the control law becomes:

$$u(t) = ly^c(t) - LX(t), \tag{7.49}$$

where $(l,\ L)$ are unknown parameters. The newest bloc diagram for the system in a closed loop is given in figure 7.2. We consider here a SISO system, this means that the gain l is a scalar and L is an array of 1 row and n columns. We assume that the direct transfer represented by the matrix D is zero. The equation 7.24 becomes, regarding the new command:

$$\begin{cases} \dot{X}(t) &= (A - BL)X(t) + ly^c(t) \\ y(t) &= CX(t) \end{cases} \tag{7.50}$$

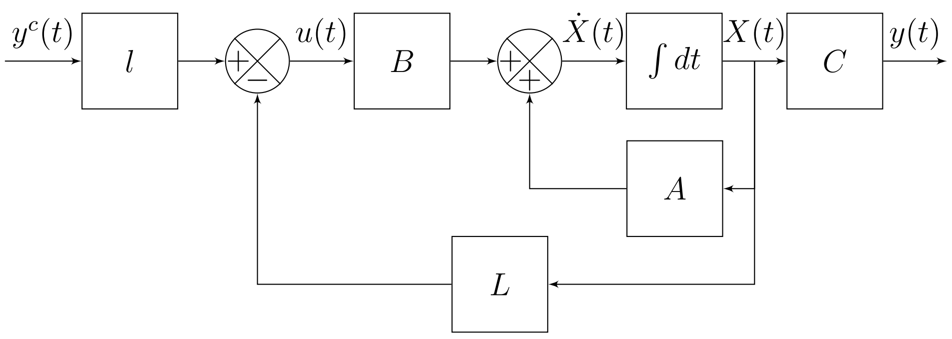


Figure 7.2: Bloc diagram in a closed loop with state feedback

The main goal of adding the vector gain is to improve the system dynamic. The pole placement technique can satisfy the desired output behavior. The user can modify easily the spectrum of the new state matrix. In others words, imposes eigenvalues of the matrix $A - BL$. The characteristic polynomial of $A - BL$ is given as below:

$$\chi_{A-BL}(\lambda) = det(\lambda I_n - (A - BL)) \tag{7.51}$$

Remark 4

We consider here an example of a second order system. Obviously, the regulator will impose two values of eigenvalues (λ_1, λ_2). The vector gain L is accessible via this system of equation:

$$\begin{cases} det(A - BL) & = & \lambda_1\lambda_2, \\ trace(A - BL) & = & \lambda_1 + \lambda_2. \end{cases}$$

7.4 State estimation

As we saw in section 7.3, the control law to be designed is defined by the state vector X. We assumed then that it is accessible and known for each $t \geq 0$. In case of unknown X for two main reasons. The vector X may not have a physical meaning or is not accessible, therefor, there is no sensors that can deliver instantaneously the values taken by the states.

We present in this section a brief but exhaustive description for a state estimator, named LUENBERGER observer. It consists on installing a set of software sensors that estimates the vector X. We assume that $\hat{X}$ is the estimated value of X. Its rate is then a function of itself, the output y, the input u.

$$\dot{\hat{X}}(t) = F\hat{X}(t) + Gu(t) + Ky(t) \tag{7.52}$$

Let us compute the error $\tilde{X}$. It consists on difference between the estimated and actual values of X.

$$\tilde{X}(t) = \hat{X}(t) - X(t) \tag{7.53}$$

The rate of $\tilde{X}$ is given by equation 7.54.

$$\dot{\tilde{X}}(t) = \dot{\hat{X}}(t) - \dot{X}(t) \tag{7.54}$$

$$\begin{aligned}\dot{\tilde{X}}(t) &= F\hat{X}(t) + Gu(t) + Ky(t) - AX(t) - Bu(t)\\ &= F\hat{X}(t) - (A - KC)X(t) + (G - B)u(t)\end{aligned} \tag{7.55}$$

$\dot{\tilde{X}}(t)$ should not be written using the input $u(t)$ in its definition. That is why we can impose $G = B$. Equation 7.55 becomes as follow:

$$\dot{\tilde{X}}(t) = F\hat{X}(t) - (A - KC)X(t) \tag{7.56}$$

Given that $\dot{\tilde{X}}(t)$ should be written entirely in function of $\tilde{x}(t)$. F can be chosen equal to $A - KC$.

$$\dot{\tilde{X}}(t) = (A - KC)\tilde{X}(t) \tag{7.57}$$

The Luenberger observer is, therefore, written as in equation 7.58.

$$\dot{\hat{X}}(t) = A\hat{X}(t) + Bu(t) + \underbrace{K(\overbrace{y(t) - C\hat{X}(t)}^{\text{Estimation error}})}_{\text{Corrective term}} \tag{7.58}$$

Remark 5

The Luenberger description of a state vector observer is also valid for the discrete system, with the same structure as in equation 7.58.

7.5 Exercises

Exercise 13

Find a state space description for the system described by the following equation

$$m\ddot{x}(t) = k(x(t) - u(t)) + f(\dot{x}(t) - \dot{u}(t)), \tag{7.59}$$

where: x is the displacement and u is the force applied to the plant. The triplet (M, k, f) denotes the mass, stiffness and the friction re-

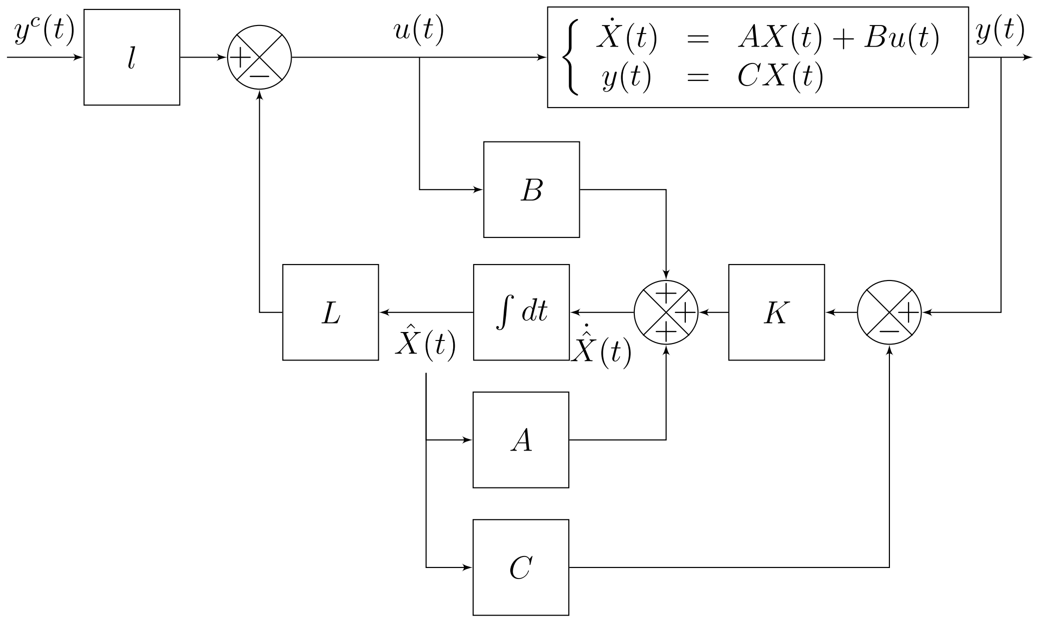


Figure 7.3: System in closed loop with LUENBERGER observer

spectively.

The system is of second order. Let us assume that $X(t)$, the state vector, is: $\begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ \dot x(t) \end{pmatrix}$. This leads to:

$$\dot x_2(t) = \frac{k}{m}x(t) + \frac{f}{m}\dot x(t) - \frac{k}{m}u(t) - \underbrace{\frac{f}{m}\dot u(t)}_{\text{Problem}} .$$

The term $\frac{f}{m}\dot u(t)$ could not be explicitly written in the state space representation. So, what we have to do is to change the second component of the vetor X. Hence, $x_2(t) = \dot x(t) + \alpha u(t)$.

$$\dot x_2(t) = \frac{k}{m}x(t) + \frac{f}{m}\dot x(t) - \frac{k}{m}u(t) - \frac{f}{m}\dot u(t) + \alpha \dot u(t).$$

To suppress $\dot u(t)$, we can choose $\alpha = \frac{f}{m}$. The system state space representation is given by equation 7.60.

$$\begin{cases} \dot X(t) & = & \begin{pmatrix} 0 & 1 \\ \frac{k}{m} & 1 \end{pmatrix} & X(t) & + & \begin{pmatrix} \frac{f}{m} \\ -\frac{k-f}{m} \end{pmatrix} & u(t) \\ x(t) & = & \begin{pmatrix} 1 & 0 \end{pmatrix} & X(t) \end{cases} \tag{7.60}$$

Exercise 14

Consider the system

$$\begin{cases} \dot{X}(t) = \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix} X(t) + \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix} u(t) \\ y(t) = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} X(t) \end{cases} \tag{7.61}$$

where α is a scalar constant.

1. How many inputs, outputs, and states does the system have?
2. Compute the poles and zeros of the system for $\alpha = 1$.
3. For what values of the constant α is the system controllable?

Code

```
>> syms alpha; A = [-1, 1; 0, -1]; B = [1, 0; 0, alpha]; ...
C = [1, 0; 1, 1];
>> N1 = size(A); N2 = size(B); N3 = size(C);
>> N2(2); % Number of inputs
>> N3(1); % Number of outputs
>> N1(2); % Number of states

>> alpha = 1;
>> eig(A) % Poles of the system
>> ???

>> clear alpha
>> ???
rank([B A*B]) = 0', 'alpha' )% System is controllable ...
% for \alpha \in \mathbb{R}\ {Result}
>> ???
```

Exercise 15

A system with one input u and two outputs y_1, y_2 is given by:

$$\begin{cases} \dot{y}_1(t) + y_1(t) + 2y_2(t) = \dot{u}(t) + 3u(t) \\ \dot{y}_2(t) + 4y_1(t) = u(t) \end{cases} \tag{7.62}$$

1. Find the transfer matrix.

2. Find a state-space representation of the system.

Exercise 16

Consider the system described by:

$$\left\{\begin{array}{lcl}\dot{X}(t) & = & \begin{pmatrix}-1 & 2\\ 0.1 & -3\end{pmatrix}X(t)+\begin{pmatrix}1\\ 1\end{pmatrix}u(t)\\ y(t) & = & \begin{pmatrix}1 & 1\end{pmatrix}X(t)\end{array}\right. \tag{7.63}$$

1. Is the system controllable and/ or observable?
2. Compute the system response for a step input.
3. Conceive a linear state feedback control for the previous system that places eigenvalues on $(-4,-5)$ and cancel the steady state error.

Code

```
>> A = [-1, 2; 0.1, -3]; B = [1; 1]; C = [1; 1];
>> rank([B A*B]) % Check the controllability
>> rank([C; C*A]) % Check the observability

>> H = tf2ss(A, B, C, 0);
>> step(H);

>> L = place(A, B, [-4, -5]; % Gain of state reaction
>> l=1/(C*inv(-A+B*L)*B); % Precompensator gain
```

Exercise 17

Consider the system described by:

$$\left\{\begin{array}{lcl}\dot{X}(t) & = & \begin{pmatrix}-1 & 0\\ 1 & -0.5\end{pmatrix}X(t)+\begin{pmatrix}-2\\ 1\end{pmatrix}u(t)\\ y(t) & = & \begin{pmatrix}0 & 1\end{pmatrix}X(t)\end{array}\right. \tag{7.64}$$

1. Determine the transfer function $H(s)=\frac{Y(s)}{U(s)}$

2. Calculate $x_1(t)$ and $x_2(t)$ when $u(t) = 0$ and $X(0^+) = \begin{pmatrix} x_1(0^+) \\ x_2(0^+) \end{pmatrix} = \begin{pmatrix} x_{1_0} \\ 0 \end{pmatrix}$.
3. Judge the controllability and observability of the system 7.64.
4. Conceive a linear state feedback control for the previous system that places eigenvalues on $(-2, -3)$ and cancel the steady state error.
5. Draw its block diagram in closed loop.

Code
```
>> A = [-1, 0; 1, -0.5]; B = [-2; 1]; C = [0, 1];
>> H = ss2tf(A, B, C, 0); % Compute the transfer function
```

Applying the Laplace transformation on equation 7.64 without considering the input $u(t)$, we obtain:

$$sX(s) - X(0^+) = AX(s) \tag{7.65}$$

So, the vector $X(t)$ is accessible via the equation 7.66.

$$X(s) = (sI_2 - A)^{-1}X(0^+) \tag{7.66}$$

Code
```
>> syms s x10
>> X = ilaplace(inv(s*eye(2)-A)*[x10; 0])
```

So the final state vector, described in the temporal domain is

$$X(t) = \begin{pmatrix} x_{1_0} e^{-t} \\ 2\, x_{1_0}(e^{\frac{-t}{2}} - e^{-t}) \end{pmatrix} \Gamma(t),$$

where $\Gamma(t)$ denotes the Heaviside step function, or the unit step function in order to ensure the causality of the calculus scheme of states in vector $X(t)$.

Code
```
>> rank([B, A*B]) % Check controllability
>> rank([C; C*A]) % Check observability

>> L = acker(A, B, [-2; -3]); % State vector feedback gain
>> l = 1/(C*inv(-A+B*L)*B);
```

Exercise 18

Give one possible state space representation and the equivalent block diagram of the following transfer function:

$$H(s) = \frac{s^3 + 25s^2 + 2.45s + 3.4}{4s^3 + 6s^2 + 10s - 2} \tag{7.67}$$

Exercise 19

Consider the following continuous state space representation:

$$\begin{cases} \dot{X}(t) &= AX(t) + Bu(t) \\ y(t) &= CX(t) + Du(t) \end{cases} \tag{7.68}$$

$$X(t) = \mathrm{e}^{A(t-t_0)}X_0 + \int_{t_0}^{t} [\mathrm{e}^{A(t-\mu)}B + D\delta(t-\mu)]u(\mu)d\mu \tag{7.69}$$

Demonstrate that the discret model corresponding to the system 7.68 can be as follows, where T_s is the sampling period:

$$\begin{cases} X_{k+1} &= FX_k + Gu_k \\ y_k &= HX_k + Du_k \end{cases} \tag{7.70}$$

$$F = \mathrm{e}^{AT_s} \qquad G = \int_0^{T_s} \mathrm{e}^{AT_s-\mu}Bd\mu \qquad H = C \tag{7.71}$$

CHAPTER 8

Adaptive Control

Truth is ever to be found in the simplicity, and not in the multiplicity and confusion of things.

Isaac Newton

Contents

ADAPTIVE CONTROL is a set of methods to determine the controllers parameters in real time. The main goal of this technique is to maintain a desired level of performances, i.e, to behave in the same way as a reference model when the parameters of the plant are not known [10]. In literature, it exists two types of adaptive control: Direct and Indirect cases[1]. These algorithms will be discussed in the coming sections.

8.1 Least Squares Method

8.1.1 An introductory example

Consider the example of a spring. Our main goal is to determine the stiffness k of this spring. The mathematical model:

$$F = k\Delta x \tag{8.1}$$

Table 8.1: Table of measurements

$\Delta x(i)$	$\Delta x(1)$	$\cdots$	$\Delta x(k)$	$\cdots$	$\Delta x(N)$
$F_{Meas}(i)$	$F_{Meas}(1)$	$\cdots$	$F_{Meas}(k)$	$\cdots$	$F_{Meas}(N)$

$$\begin{aligned} F_{Meas}(i) &= F(i) + e_i &(8.2)\\ &= k\Delta x(i) + e_i, &(8.3)\end{aligned}$$

where $F(i)$ denotes the unknown real value of the force applied to the spring. In order to estimate the value of the stiffness value of k, we have to consider the quadratic criterion:

$$\begin{aligned} J &= \sum_{i=1}^{N} e_i^2 &(8.4)\\ &= \sum_{i=1}^{N} (F_{Meas}(i) - k\Delta x(i))^2 &(8.5)\end{aligned}$$

[1]See: http://en.wikipedia.org/wiki/Adaptive_control

$$\frac{\partial J}{\partial k} = 0 \tag{8.6}$$

$$\begin{aligned}
\Longrightarrow 2\sum_{i=1}^{N}[F_{Meas}(i) - k\Delta x(i)]\sum_{i=1}^{N}\frac{\partial[F_{Meas}(i) - k\Delta x(i)]}{\partial k} &= 0\\
\Longrightarrow \sum_{i=1}^{N}[F_{Meas}(i) - k\Delta x(i)]\sum_{i=1}^{N}\Delta x(i) &= 0
\end{aligned} \tag{8.7}$$

$$\sum_{i=1}^{N}[F_{Meas}(i)\Delta x(i)] = k\sum_{i=1}^{N}(\Delta x(i))^2 \tag{8.8}$$

$$\boxed{\hat{k} = \frac{\sum_{i=1}^{N} F_{Meas}(i)\Delta x(i)}{\sum_{i=1}^{N}(\Delta x(i))^2} \tag{8.9}}$$

8.1.2 Second example

This example consists on determining the unknown couple (y_0, v_0) of a mobile solid. We assume that the trajectory is linear and . The mathematical model that relates the position y to time t is given by this equation:

$$y = y_0 + vt \tag{8.10}$$

Table 8.2: Measurements of position y

t_i	t_1	$\cdots$	t_k	$\cdots$	t_N
$y_{Meas}(i)$	$y_{Meas}(1)$	$\cdots$	$y_{Meas}(k)$	$\cdots$	$y_{Meas}(N)$

$$\begin{aligned}
y_{Meas}(i) &= y_i + e_i &(8.11)\\
&= y_0 + vt_i + e_i, &(8.12)
\end{aligned}$$

where y_i denotes the unknown real value of the position y at instant t_i. In order to estimate the values taken by the couple $(\ v,\ \ y_0\ )$, we have to consider the quadratic criterion:

$$\begin{aligned}
J &= \sum_{i=1}^{N} e_i^2 &(8.13)\\
&= e^T e &(8.14)
\end{aligned}$$

The vector e is formed by the set of e_i, $e = \begin{pmatrix} e_1 & \cdots & e_N \end{pmatrix}^T$.

$$\frac{\partial J}{\partial \begin{pmatrix} v, & y_0 \end{pmatrix}} = 0 \tag{8.15}$$

To find a well estimated value to the couple $\begin{pmatrix} v, & y_0 \end{pmatrix}$, we have to compute the absolute minimum of J. We will exhibit in detail the results for a multidimensional case in subsection 8.1.3.

8.1.3 Non-recursive Method

$$y = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{8.16}$$

$$y_k = \theta_1 x_{(1,k)} + \theta_2 x_{(2,k)} + \cdots + \theta_n x_{(n,k)} + e_k \tag{8.17}$$

Later,we denote by θ the vector $\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$. The function y_k becomes:

$$y_k = \underbrace{[x_{1,k},\, x_{2,k},\, \cdots,\, x_{n,k}]}_{h_k^T}\theta + e_k \tag{8.18}$$

We assume that we have N measurements for the scalar y. Then we can write the equation 8.18 as follows:

$$Y = H\theta + e, \tag{8.19}$$

where $Y^T = [y_1, y_2, \cdots, y_N]$, $H = \begin{bmatrix} h_1^T \\ h_2^T \\ \vdots \\ h_N^T \end{bmatrix}$ and $e^T = [e_1, e_2, \cdots, e_N]$. To compute the approximated value of θ, we can consider the following quadratic criterion:

$$J = \sum_{i=0}^{N} e_i^2 = e^T e \tag{8.20}$$

The best well estimated value of $\hat{\theta}$ corresponds to the absolute minimum of J. This leads to calculate the gradient of J with respect to θ, given by equation 8.21.

$$\frac{\partial J}{\partial \theta} = \frac{\partial (e^T e)}{\partial \theta} \tag{8.21}$$

$$\frac{\partial (e^T e)}{\partial \theta} = 2(\frac{\partial e}{\partial \theta})^T e \tag{8.22}$$

given $e = Y - H\theta$, $\frac{\partial e}{\partial \theta}$ becomes:

$$\frac{\partial e}{\partial \theta} = -H \tag{8.23}$$

$$\frac{\partial J}{\partial \theta} = 2(-H)^T(Y - H\theta) \tag{8.24}$$
$$= 0 \tag{8.25}$$

The estimation algorithm is, then, given by equation 8.26.

$$\boxed{\hat{\theta} = (H^T H)^{-1} H^T Y} \tag{8.26}$$

Exercise 20

Consider the following transfer function:

$$H(s) = \frac{K}{1 + \tau s} \tag{8.27}$$

We are interested here in estimating the couple (K, τ) based on the table 8.3 which summarizes the response of both real and imaginary parts of the function H to a sinusoidal input.

1. Using the real part of $H(\jmath w)$, estimate $\theta = [K, \tau]^T$.
2. Using the imaginary part of $H(\jmath w)$, estimate θ.
3. Compare results and deduce the reason of building the estimation algorithm from the two responses.

Table 8.3: Table of measurements

w	0.03	0.072	0.15	0.35	0.8
Real part of $H(\jmath w)$	1.17	1.06	0.77	0.3	0.072
Imaginary part of $H(\jmath w)$	-0.18	-0.38	-0.57	-0.52	-0.28

$$\begin{aligned} H(\jmath w) &= \frac{K}{1 + \jmath \tau w} \\ &= \frac{K}{1 + (\tau w)^2}(1 - \jmath \tau w) \end{aligned} \tag{8.28}$$

Thereafter, $\Re_i$ and $\Im_i$ denotes respectively the real and imaginary parts of $H(\jmath w)$ relative to the pulsation w_i.

$$\Re_i = \frac{K}{1 + (\tau w_i)^2} \tag{8.29}$$
$$\Im_i = \frac{K}{1 + (\tau w_i)^2} \tau w_i \tag{8.30}$$

From the equations 8.29 and 8.30, θ will be accessible via these two new equations:

$$\Re_i = [1,\ -\Re_i w_i^2] \underbrace{\begin{bmatrix} K \\ \tau^2 \end{bmatrix}}_{\theta_1} \tag{8.31}$$

$$\Im_i = [1,\ -\Im_i w_i^2] \underbrace{\begin{bmatrix} K\tau \\ \tau^2 \end{bmatrix}}_{\theta_2} \tag{8.32}$$

Notice that K and τ are given by this two equations:

$$\hat{K} = \hat{\theta}_1(1), \qquad \hat{\tau} = \sqrt{\hat{\theta}_1(2)} > 0 \tag{8.33}$$

$$\hat{K} = \frac{\hat{\theta}_2(1)}{\hat{\tau}}, \qquad \hat{\tau} = \sqrt{\hat{\theta}_2(2)} > 0 \tag{8.34}$$

Code

```
>> W = [0.03; 0.072; 0.15; 0.35; 0.8];
>> Y1 = [1.17; 1.06; 0.77; 0.3; 0.072];
>> Y2 = [-0.18; -0.38; -0.57; -0.52; -0.28];
>> H1 = [ones(length(W), 1), -Y1.*(W.^2)];
>> H2 = [ones(length(W), 1), -Y2.*(W.^2)];
>> Theta1 = inv(H1'*H1)*H1'*Y1;
>> HatK1 = Theta1(1); % Approximated value of K
>> sqrt(Theta1(2));
>> HatTau1 = ans(ans>0); % Only use the positive value
>> Theta2 = inv(H2'*H2)*H2'*Y2;
>> sqrt(Theta2(2));
>> HatTau2 = ans(ans>0);
>> HatK2 = Theta2(1)/ HatTau2;
```

The selected pair of K and τ are those who verify both equations 8.33 and 8.34.

In general, a complex quantity is expressed as in equation 8.35.

$$H(jw) = \underbrace{|H(jw)|\cos(\phi)}_{\Re} + j\underbrace{|H(jw)|\sin(\phi)}_{\Im} \tag{8.35}$$

So, building an estimation algorithm based only on $\Re$ or $\Im$ quantities is not sufficient, due to the following causes given in equation 8.36.

$$\cos(\phi) = \cos(-\phi), \qquad \sin(\phi) = \sin(\pi - \phi) \tag{8.36}$$

By combining the two parts, we guarantee the unicity of found solutions.

Exercise 21

Consider the discrete state space equation 8.37:

$$\begin{cases} X_{k+1} &= \begin{pmatrix} a & 0 \\ b & 1 \end{pmatrix} X_k + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u_k \\ y_k &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X_k \end{cases} \tag{8.37}$$

Testing the discrete process have yielded the values indicated on table 8.4.

1. Estimate the values of the couple $(a,\, b)$ assumed constants.

Table 8.4: Table of measurements

k	0	1	2	3	4	5	6
u_k	-1	-1	-1	-1	-1	-1	-1
y_k	$\begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ -0.28 \end{pmatrix}$	$\begin{pmatrix} 0.26 \\ -0.3 \end{pmatrix}$	$\begin{pmatrix} 0.56 \\ -0.5 \end{pmatrix}$	$\begin{pmatrix} 0.7 \\ -0.63 \end{pmatrix}$	$\begin{pmatrix} 0.45 \\ 0.34 \end{pmatrix}$	$\begin{pmatrix} 0.6 \\ 0.6 \end{pmatrix}$

8.1.4 Recursive method

Let us consider the equation 8.26 relative to $k+1$ measurements:

$$\hat{\theta}_{k+1} = (H_{k+1}^T H_{k+1})^{-1} H_{k+1}^T Y_{k+1} \tag{8.38}$$

In equation 8.38, assume that $P_k^{-1} = H_{k+1}^T H_{k+1}$ and $b_{k+1} = H_{k+1}^T Y_{k+1}$.

Let us remember the expression of H_{k+1}

$$b_{k+1} = H_{k+1}^T Y_{k+1} \tag{8.39}$$

$$= [h_1,\,\cdots,\,h_k,\,h_{k+1}] \begin{bmatrix} y_1 \\ \vdots \\ y_k \\ y_{k+1} \end{bmatrix} \tag{8.40}$$

$$= \left[\underbrace{\begin{bmatrix} x_{1,\,1} \\ x_{2,\,1} \\ \vdots \\ x_{n,\,1} \end{bmatrix}}_{h_1}, \begin{bmatrix} \cdots \\ \cdots \\ \vdots \\ \cdots \end{bmatrix}, \underbrace{\begin{bmatrix} x_{1,\,k} \\ x_{2,\,k} \\ \vdots \\ x_{n,\,k} \end{bmatrix}}_{h_k}, \underbrace{\begin{bmatrix} x_{1,\,k+1} \\ x_{2,\,k+1} \\ \vdots \\ x_{n,\,k+1} \end{bmatrix}}_{h_{k+1}} \right] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \\ y_{k+1} \end{bmatrix} \tag{8.41}$$

$$= \underbrace{\left[\begin{bmatrix} x_{1,\,1} \\ x_{2,\,1} \\ \vdots \\ x_{n,\,1} \end{bmatrix}, \begin{bmatrix} \cdots \\ \cdots \\ \vdots \\ \cdots \end{bmatrix}, \begin{bmatrix} x_{1,\,k} \\ x_{2,\,k} \\ \vdots \\ x_{n,\,k} \end{bmatrix} \right]}_{H_k^T} \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}}_{Y_k} + \underbrace{\begin{bmatrix} x_{1,\,k+1} \\ x_{2,\,k+1} \\ \vdots \\ x_{n,\,k+1} \end{bmatrix}}_{h_{k+1}} y_{k+1} \tag{8.42}$$

We deduce from equation 8.42 that $b_{k+1} = H_k^T Y_k + h_{k+1} y_{k+1}$. Let us now examine the vector $\hat{\theta}_{k+1}$ in equation 8.38:

$$\hat{\theta}_{k+1} = (H_{k+1}^T H_{k+1})^{-1} (\overbrace{H_k^T Y_k}^{b_k} + h_{k+1} y_{k+1}) \tag{8.43}$$

$$P_{k+1}^{-1} = H_{k+1}^T H_{k+1} = [h_1,\,\cdots,\,h_k,\,h_{k+1}] \begin{bmatrix} h_1^T \\ \vdots \\ h_k^T \\ h_{k+1}^T \end{bmatrix} \tag{8.44}$$

In order to establish an explicit expression of P_{k+1}^{-1}, let us examine its i^{th} and the j^{th} column, where $i,\,j = 1,\,\cdots,\,k,\,k+1$.

$$P_{k+1}^{-1}(i,\,j) = x_{i,\,1} x_{j,\,1} + \cdots + x_{i,\,k} x_{j,\,k} + x_{i,\,k+1} x_{j,\,k+1} \tag{8.45}$$

$$= \sum_{l=1}^{k+1} x_{i,\,l} x_{j,\,l} \tag{8.46}$$

$$= \sum_{l=1}^{k+1} h_l h_l^T (i,\,j) \tag{8.47}$$

Hence, the adaptive gain will be transformed as shown below in equation 8.48.

$$P_{k+1}^{-1} = \sum_{i=0}^{k+1} h_i h_i^T \tag{8.48}$$

$$= \underbrace{\sum_{i=0}^{k} h_i h_i^T}_{P_k^{-1}} + h_{k+1}h_{k+1}^T \tag{8.49}$$

Since $P_{k+1}^{-1} = P_k^{-1} + h_{k+1}h_{k+1}^T$, we can use the following lemma for matrix inversion.

Lemma 8.1.

$$(P_k^{-1} + h_{k+1}h_{k+1}^T)^{-1} = P_k - P_k h_{k+1}(1 + h_{k+1}^T P_k h_{k+1})^{-1} h_{k+1}^T P_k \tag{8.50}$$

Given the equation 8.43, the newest value of the matrix P_{k+1}^{-1} is computed as follows:

$$\begin{aligned}
\hat{\theta}_{k+1} &= P_{k+1}(b_k + h_{k+1}y_{k+1}) \\
&= P_{k+1}b_k + \underbrace{P_{k+1}h_{k+1}}_{K_{k+1}} y_{k+1} \\
&= \underbrace{P_k b_k}_{\hat{\theta}_k} + K_{k+1}y_{k+1} - P_k h_{k+1}(1 + h_{k+1}^T P_k h_{k+1})^{-1} h_{k+1}^T \underbrace{P_k b_k}_{\hat{\theta}_k} \\
&= \hat{\theta}_k + K_{k+1}y_{k+1} - \frac{P_k}{1 + h_{k+1}^T P_k h_{k+1}} h_{k+1}h_{k+1}^T \hat{\theta}_k
\end{aligned} \tag{8.51}$$

$$P_{k+1}^{-1} = P_k^{-1} + h_{k+1}h_{k+1}^T \tag{8.52}$$

Multiply the left hand of the previous equations by P_k and the right hand by P_{k+1}, we obtain:

$$P_k = P_{k+1} + P_k h_{k+1}h_{k+1}^T P_{k+1} \tag{8.53}$$

This means that matrix P_{k+1} becomes as given in equation 8.54:

$$P_{k+1} = P_k - P_k h_{k+1}h_{k+1}^T P_{k+1} \tag{8.54}$$

Making identification between the equations 8.51 and 8.54, we deduce that:

$$P_{k+1} = \frac{P_k}{1 + h_{k+1}^T P_k h_{k+1}} \tag{8.55}$$

$$\hat{\theta}_{k+1} = \hat{\theta}_k + K_{k+1}y_{k+1} - \underbrace{P_{k+1}h_{k+1}}_{K_{k+1}} h_{k+1}^T \hat{\theta}_k \tag{8.56}$$

$$= \hat{\theta}_k + K_{k+1}(y_{k+1} - h_{k+1}^T \hat{\theta}_k) \tag{8.57}$$

The recursive algorithm is done in three repetitive iterations in addition to the initialization step: We have to give initial values for both $\hat{\theta}_0$ and P_0.

1. The first step is to compute the adaptive gain P_{k+1}, given in equation 8.58.

$$P_{k+1} = P_k - \frac{P_k h_{k+1} h_{k+1}^T P_k}{1 + h_{k+1}^T P_k h_{k+1}} \tag{8.58}$$

2. We calculate then the corrective gain K_{k+1} as in equation 8.59.

$$K_{k+1} = P_{k+1} h_{k+1} \tag{8.59}$$

3. The ultimate step is to compute, according to equation 8.60, the new approximated value of $\hat{\theta}_{k+1}$

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \underbrace{K_{k+1}(\overbrace{y_{k+1} - h_{k+1}^T \hat{\theta}_k}^{\text{Estimation error}})}_{\text{Corrective term}} \tag{8.60}$$

8.2 Generalized Least Squares Method

As we saw in the previous section, the least squares method is based upon minimizing the quadratic criterion. This approach for estimating a parameter vector is only valid in case of a white Gaussian noise which is not the case of all experimentation environments. If the noise is colored, then the last exposed algorithm is biased. In this stage, we may have a callback to the statistics properties of the algorithm. The new conceived algorithm has to have two main properties, where $\tilde{\theta}$ denotes the estimation error:

$$\mathbb{E}[\hat{\theta}] = \mathbb{E}[\theta] \tag{8.61}$$

$$var(\tilde{\theta}) = \mathbb{E}[(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T] \to 0 \tag{8.62}$$

Generally, the variance of $\tilde{\theta}$ is a function of measurement's number and parameters.

8.2.1 Hypothesis

Consider an ARMAX/ CARMA model. Its representative equation can be written in the following form:

$$\underbrace{A(q^{-1})y_k}_{\text{Auto Regressive}} = \underbrace{B(q^{-1})u_k}_{\text{eXogenous}} + \underbrace{C(q^{-1})e(k)}_{\text{Moving Average}} \tag{8.63}$$

The sequence is assumed to be a Gaussian random vector. Having a zero mean $\mathbb{E}[e] = 0$ and covariance matrix equal to identity $\mathbb{E}[e^T e] = I$. The variance is assumed to be unitary, if it is not the case, consider a scalar λ. The considered system is stable and the polynomials A, C and B, C are pairwise coprimes.

The existence of the polynomial C adds to the noise series e a color. Then, the algorithm given in sub-subsection **??** is no longer valid. In order to be able to use the Least Squares Method, we have to whiten the sequence $e(k)$. First of all, think of dividing the two members of the whole equation 8.63 by the origin of coloration, the term C.

Do not forget that this equation is the mathematical description of the system, so the division to be held by the polynomial C consists on physically, filtering both the input and the output.

The system can be transformed, then into the equation 8.64.

$$A(q^{-1})\underbrace{\frac{y_k}{C(q^{-1})}}_{y_F(k)} = B(q^{-1})\underbrace{\frac{u_k}{C(q^{-1})}}_{u_F(k)} + e(k) \tag{8.64}$$

$$A(q^{-1})y_F(k) = B(q^{-1})u_F(k) + e(k) \tag{8.65}$$

Finally, you can just apply the algorithm given in subsection 8.2.2.

8.2.2 Algorithm

1. Take a filter
$$\hat{F}_i(q^{-1}) = 1 + \hat{f}_{(i,1)}q^{-1} \cdots + \hat{f}_{(i,n)}q^{-n} \tag{8.66}$$
2. Compute the filtered input and output
$$u_i^F(k) = \hat{F}_i(q^{-1})u(k) \tag{8.67}$$
$$y_i^F(k) = \hat{F}_i(q^{-1})y(k) \tag{8.68}$$
3. Estimate the vector $\hat{\theta}_k$ using the least squares method.
4. Compute the residuals w
$$w_i(k) = \hat{A}_i(q^{-1})y(k) - \hat{B}_i(q^{-1})u(k) \tag{8.69}$$
5. Estimate the coefficients of the filter 8.66 such that:
$$\hat{F}_i(q^{-1})w_i(k) = e(k) \tag{8.70}$$
6. Return to step 2.

8.3 Instrumental Variable

8.3.1 Non-recursive Method

As indicated in the subsection 8.1.3, we supposed that all parameters x_i are linearly independents and known exactly. Only the output y_k is corrupted by noise measurements.

$$y_k = \theta_1 x_{(1,k)} + \theta_2 x_{(2,k)} + \cdots + \theta_n x_{(n,k)} + e_k \tag{8.71}$$

In this case, for a regression written as described in equation 8.71, the Ordinary Least Squares method given by the equation 8.79 yields true parameters. However, in the presence of a large amount of noise, the Ordinary Least Squares method is biased and the estimate $\hat{\theta}$ is not consistent.

$$\hat{\theta} = (H^T H)^{-1} H^T Y \tag{8.72}$$

Let us suppose that the actual values of $\hat{x}_i$ is corrupted by measurement noise v_i, that is:

$$x_i = \hat{x}_i + v_i, \tag{8.73}$$

where v_i is assumed to be a strictly stationary stochastic process with zero mean.

Equation 8.71 becomes:

$$y_k = \theta_1(\hat{x}_{(1,k)} + v_{(1,k)}) + \theta_2(\hat{x}_{(2,k)} + v_{(2,k)}) + \cdots + \theta_n(\hat{x}_{(n,k)} + v_{(n,k)}) + e_k \tag{8.74}$$

Define:

$$h_k^T = \begin{bmatrix} x_{1,k} & x_{2,k} & \cdots & x_{n,k} \end{bmatrix}; \tag{8.75}$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}; \tag{8.76}$$

$$\epsilon_k = \theta_1 v_{(1,k)} + \theta_2 v_{(2,k)} + \cdots + \theta_n v_{(n,k)}. \tag{8.77}$$

Then equation 8.75 can be expressed as:

$$y_k = h_k^T \theta + \epsilon_k. \tag{8.78}$$

It follows from [11] that in such case, the OLS estimate is not consistent, because ϵ is now correlated with h^T. One solution consists on using the Instrumental Variable (IV) method.

Proposition 8.2. For the system given in equation 8.78, if the instrumental variable matrix Φ is chosen to satisfy the following two limiting properties:

i. $\lim_{N\to\infty} \frac{1}{N}\Phi^T H$ exists and is not singular;

ii. $\lim_{N\to\infty} \frac{1}{N}\Phi^T \epsilon = 0$,

then the estimate given by:

$$\hat{\theta} = (\Phi^T H)^{-1}\Phi^T Y \tag{8.79}$$

is consistent.

8.3.2 Recursive method

Given a linear regression:

$$y_k = h_k^T\theta + \epsilon_k, \tag{8.80}$$

the IV estimate is given by:

$$\hat{\theta}_k = \left[\sum_{k=1}^{N}\right]^{-1}\left[\sum_{k=1}^{N}\right]. \tag{8.81}$$

1. The first step is to compute the adaptive gain P_{k+1}, given in equation 8.82.

$$P_{k+1} = P_k - \frac{P_k\phi_{k+1}h_{k+1}^T P_k}{1 + h_{k+1}^T P_k \phi_{k+1}} \tag{8.82}$$

2. We calculate then the corrective gain K_{k+1} as in equation 8.83.

$$K_{k+1} = P_{k+1}\phi_{k+1} \tag{8.83}$$

3. The ultimate step is to compute, according to equation 8.84, the new approximated value of $\hat{\theta}_{k+1}$

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \underbrace{K_{k+1}(\overbrace{y_{k+1} - h_{k+1}^T\hat{\theta}_k}^{\text{Estimation error}})}_{\text{Corrective term}} \tag{8.84}$$

Given that the system can be approximated by a (FOPDT) model, the transient output to a step input with amplitude u_∞ with initial condition y_0 can be described as in (8.85).

$$y(t) = \begin{cases} y_0 e^{-\frac{t}{T}} + \omega(t), & \forall t \leq \tau, \\ K u_\infty(1 - e^{-\frac{t-\tau}{T}}) + y_0 e^{-\frac{t}{T}} + \omega(t), & \forall t \geq \tau. \end{cases} \tag{8.85}$$

Where $\omega(t)$ is assumed to be a white noise and $y_0 = 0$. Integrating $y(t)$ in (8.85), $\forall t \geq \tau$, yields:

$$\begin{aligned} \int_0^t y &= Ku_\infty \int_\tau^t (1 - \mathrm{e}^{-\frac{t-\tau}{T}}) + \int_0^t \omega, \\ &= Ku_\infty (t - \tau - T(1 - \mathrm{e}^{-\frac{t-\tau}{T}})) + \int_0^t \omega. \end{aligned} \tag{8.86}$$

Considering (8.85), the term $1 - \mathrm{e}^{-\frac{t-\tau}{T}}$ can be written as in (8.87) given that $y_0 = 0$.

$$1 - \mathrm{e}^{-\frac{t-\tau}{T}} = \frac{y(t) - \omega}{Ku_\infty}, \qquad \forall t \geq \tau. \tag{8.87}$$

(8.86) can be transformed into (8.88).

$$\underbrace{\int_0^t y}_{\psi(t)} = Ku_\infty (t - \tau) - Ty(t) + \underbrace{T\omega + \int_0^t \omega}_{\delta(t)}, \qquad \forall t \geq \tau. \tag{8.88}$$

The linear regression can be written as indicated in (8.89).

$$\psi(t) = \begin{pmatrix} tu_\infty & -u_\infty & -y(t) \end{pmatrix} \begin{pmatrix} K \\ K\tau \\ T \end{pmatrix} + \delta(t), \quad \forall t \geq \tau. \tag{8.89}$$

Gathering the sampled process response for $t \geq \tau$ yields a system of linear equations:

$$\Psi = H\theta + \Delta, \tag{8.90}$$

where:

$$\Psi = \begin{bmatrix} \psi_m \\ \vdots \\ \psi_{m+k} \\ \vdots \\ \psi_{m+N} \end{bmatrix}, \quad H = \begin{bmatrix} mT_s u_\infty & -1 & -y_m \\ \vdots & \vdots & \vdots \\ (m+k)T_s u_\infty & -1 & -y_{m+k} \\ \vdots & \vdots & \vdots \\ (m+N)T_s u_\infty & -1 & -y_{m+N} \end{bmatrix},$$

$$\Delta = \begin{bmatrix} \delta_m \\ \vdots \\ \delta_{m+k} \\ \vdots \\ \delta_{m+N} \end{bmatrix}, \qquad \theta = \begin{bmatrix} K \\ K\tau \\ T \end{bmatrix}.$$

We denote by f_k the value of f at instant kT_s, i.e., $f_k = f(t = kT_s)$ where T_s is the sampling period. (8.88) is valid only for $t \geq \tau$, then m has to be chosen greater or equal to $\frac{\tau}{T_s}$. In the noise-free testing environment, using the least-squares method as in (8.91) yields true parameters.

$$\hat{\theta} = (H^T H)^{-1} H^T \Psi. \tag{8.91}$$

However, the process output as shown in figure ?? is corrupted by a large amount of noise, $\delta(t) = T\omega(t) + \int_0^t \omega$. The ordinary least squares method is then biased and the estimate $\hat{\theta}$ is not consistent because Δ is now correlated with H. One solution consists on using the Instrumental Variable (IV) method [?, ?].

The matrix Φ can be chosen as in (8.92).

$$\Phi = \begin{bmatrix} mT_s & -1 & \frac{1}{mT_s} \\ \vdots & \vdots & \vdots \\ (m+k)T_s & -1 & \frac{1}{(m+k)T_s} \\ \vdots & \vdots & \vdots \\ (m+N)T_s & -1 & \frac{1}{(m+N)T_s} \end{bmatrix}. \tag{8.92}$$

Lemma 8.3. For the system given by (8.89) and (8.90), the matrix Φ in (8.92) is an instrumental variable matrix which satisfies the condition in proposition **??**.i.

Proof. Computing $\Phi^T H$ leads to:

$$\Phi^T H = \begin{bmatrix} \sum_{k=0}^N u_\infty((m+k)T_s)^2 & -\sum_{k=0}^N (m+k)T_s & -\sum_{k=0}^N (m+k)T_s y_{m+k} \\ -\sum_{k=0}^N u_\infty(m+k)T_s & (N+1) & \sum_{k=0}^N y_{m+k} \\ u_\infty(N+1) & -\sum_{k=0}^N \frac{1}{(m+k)T_s} & -\sum_{k=0}^N \frac{y_{m+k}}{(m+k)T_s} \end{bmatrix}. \tag{8.93}$$

Since the system described in section **??** is stable, the output value $y(t)$ is then bounded.

$$\lim_{N\to\infty} \frac{1}{N}\Phi^T H = \begin{bmatrix} \mathcal{O}(N) & \mathcal{O}(1) & \mathcal{O}(1) \\ \mathcal{O}(1) & \mathcal{O}(1) & \mathcal{O}(\frac{1}{N}) \\ \mathcal{O}(1) & \mathcal{O}(\frac{1}{N^2}) & \mathcal{O}(\frac{1}{N^2}) \end{bmatrix}. \tag{8.94}$$

The determinant of the above matrix is as same order as N^0, i.e. $\mathcal{O}(1)$. Therefore,

$$det(\lim_{N\to\infty} \frac{1}{N}\Phi^T H) = \mathcal{O}(1), \tag{8.95}$$

and the inverse of

$$\lim_{N\to\infty} \frac{1}{N}\Phi^T H$$

exists. □

Figure **??** shows the trajectories of estimated parameters. The resulting estimated model is written as:

$$\hat{H}(s) \approx \frac{0.5638e^{-0.978s}}{1+1.33s}. \tag{8.96}$$

As seen above, this method allows a direct and robust identification of a first order model with dead-time from a step response output, corrupted by measurement noise. The main drawback of such algorithm is that the process output has to be monitored for a period, called the "listening period", where the system is not yet fully identifiable. The first sample of the process output y_m should not be taken until mT_s is greater than the unknown delay τ. A solution developed in [?, ?] consists on determining the noise band B_n and starting at $t = mT_s$ assuming that the process response $y(t \geq mT_s)$ is greater than the double of B_n:

$$y(t \geq mT_s) \geq 2B_n. \tag{8.97}$$

In this case, all estimated parameters can be maintained equal to zero till the condition in (8.97) is verified.

To avoid the issue concerning the implementation of the algorithm, a new method has been developed in [?] to estimate a first order plus time-delay model from step response using a new linear regression equation. This approach uses filtered signals obtained from the plant output. Hence, Noise sensitivity is reduced and the algorithm yields exact formulas for the unknown parameters.

8.4 Case of variable parameters

$$P_k^{-1} = \lambda_{1_k} P_{k-1}^{-1} + \lambda_{2_k} h_k h_k^T \tag{8.98}$$

$$P_k = \frac{1}{\lambda_{1_k}} \left[P_{k-1} - \frac{P_{k-1} h_k h_k^T P_{k-1}}{\frac{\lambda_{1_k}}{\lambda_{2_k}} + h_k^T P_{k-1} h_k} \right] \tag{8.99}$$

CHAPTER 9

Nonlinear Least Squares Curve-Fitting Problems

****.

***

Contents

IN FITTING a functional relation $\hat{y}$ of an independent variable t and which maps a parameter vector $x \in \mathrm{R}^p$, to a set of measurement data vector $y \in \mathrm{R}^n$, it is convenient to minimize the sum of the squared weighted residuals $\left(\dfrac{y_i - \hat{y}_i}{\sigma_{ii}^2}\right)^2$, between the measured data $y_i(x)$ and estimated functional $\hat{y}_i(x)$. The scalar, as shown in equation (9.1), is called the CHI-SQUARED error criterion.

$$\chi^2(x) = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{\sigma_{ii}^2} \right)^2 \tag{9.1}$$

$$= (y - \hat{y})^T \Sigma_x^{-1} (y - \hat{y}). \tag{9.2}$$

The term χ^2 measures the degree of how faithful are measured data to estimated model. The weighting matrix Σ_x^{-1} is the covariance matrix for the measured vector x. If the elements of x are not correlated between them, matrix Σ_x^{-1} is then diagonal with $\Sigma_{x_{ii}}^{-1} = \frac{1}{\sigma_i^2}$, where $\sigma_i^2 = cov(x_i, x_i)$.

$$\Sigma_x^{-1} = \begin{pmatrix} \sigma_{11}^2 & 0 & \cdots & \cdots & 0 \\ 0 & \sigma_{22}^2 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & \sigma_{pp}^2 \end{pmatrix}. \tag{9.3}$$

Functional $\hat{y}$ is non linear on vector x, the curve-fitting scheme has then to be carried out iteratively. The goal of each iteration is to find the most suitable perturbation δ_x to parameters x that minimizes χ^2. The minimum is found by solving a weighted least squares problem.

9.1 The Gradient Descent Method

The gradient of the chi-squared objective function with respect to the parameters x is

$$\frac{\partial \chi^2}{\partial x} = (y - \hat{y})^T \Sigma_x^{-1} \frac{\partial (y - \hat{y})}{\partial x} \tag{9.4}$$

$$= -(y - \hat{y})^T \Sigma_x^{-1} \frac{\partial \hat{y}}{\partial x} \tag{9.5}$$

Consider

$$\mathbf{J} = \frac{\partial \hat{y}}{\partial x}, \tag{9.6}$$

J is the $n \times p$ Jacobian matrix. It represents the local sensitivity of the function $\hat{y}$ to variation in parameters x. For notational simplicity, equation (9.5) becomes

$$\frac{\partial \chi^2}{\partial x} = -(y - \hat{y})^T \Sigma_x^{-1} \mathbf{J} \tag{9.7}$$

The parameter update δ_x that moves the parameters in the direction of steepest descent is given by equation (9.8)

$$\delta_x = \alpha \mathbf{J}^T \Sigma_x^{-1} (y - \hat{y}), \tag{9.8}$$

9.2 The Gauss-Newton Method

$$\begin{aligned}\hat{y}(x+\delta_x) &\approx \hat{y}(x) + \left(\frac{\partial \hat{y}}{\partial x}\right)\delta_x && (9.9)\\ &\approx \hat{y}(x) + \mathbf{J}\delta_x && (9.10)\end{aligned}$$

$$\left(\mathbf{J}^T\Sigma_x^{-1}\mathbf{J}\right)\delta_x = \mathbf{J}^T\Sigma_x^{-1}(y-\hat{y}). \qquad (9.11)$$

9.3 The Levenberg-Marquardt Method

The technique developed in this section, is commonly regarded as an alternation between the steepest descent and the Gauss-Newton method depending on how far being from the minimum. This procedure updates the value of the parameter vector perturbation δ_x depending on the value of a positive scalar λ as given in equation 9.12.

$$\left(\mathbf{J}^T\Sigma_x^{-1}\mathbf{J} + \lambda\,\mathtt{I}\right)\delta_x = \mathbf{J}^T\Sigma_x^{-1}(y-\hat{y}). \qquad (9.12)$$

λ is called the damping term. The strategy of altering the diagonal elements of matrix $\mathbf{J}^T\mathbf{J}$ is commonly referred to as damping.

If the perturbation found in equation (9.12) leads to a reduction in the error $(y-\hat{y})$ when updating the parameter vector $x+\delta_x$, the process is updated with reduced damping term λ. Otherwise, the update is rejected, λ is increased and the process repeats until a value of δ_x that decreases error is found. The Levenberg-Marquardt approach for non-linear curve fitting problem is adaptive because it controls its own damping factor. It is capable to alternate between a slow gradient descent procedure when being far from the optimal solution and the Gauss-Newton method when reaching the optimal's neighborhood.

One other suggestion to locate the suitable δ_x when updating $x+\delta_x$ can be found in [12] and is given by equation (9.13).

$$\left(\mathbf{J}^T\Sigma_x^{-1}\mathbf{J} + \lambda\,\mathtt{diag}(\mathbf{J}^T\Sigma_x^{-1}\mathbf{J})\right)\delta_x = \mathbf{J}^T\Sigma_x^{-1}(y-\hat{y}). \qquad (9.13)$$

The table below summarizes the update law for δ_x when applying the previously exhibited methods.

Method	Perturbation
Gradient Descent	$\delta_x = \alpha\mathbf{J}^T\Sigma_x^{-1}(y-\hat{y})$
Gauss-Newton	$\left(\mathbf{J}^T\Sigma_x^{-1}\mathbf{J}\right)\delta_x = \mathbf{J}^T\Sigma_x^{-1}(y-\hat{y})$
Levenberg-Marquardt	$\left(\mathbf{J}^T\Sigma_x^{-1}\mathbf{J} + \lambda\,\mathtt{diag}(\mathbf{J}^T\Sigma_x^{-1}\mathbf{J})\right)\delta_x = \mathbf{J}^T\Sigma_x^{-1}(y-\hat{y})$

Backward difference

$$\begin{aligned}J_{lp} &= \frac{\partial \hat{y}_l}{\partial x_p} && (9.14)\\ &= \frac{\hat{y}(t_l,\,x) - \hat{y}(t_l,\,x+\delta_{x_p}}{||\delta_{x_p}||} && (9.15)\end{aligned}$$

Forward difference

$$J_{lp} = \frac{\partial \hat{y}_l}{\partial x_p} \tag{9.16}$$

$$= \frac{\hat{y}(t_l,\, x + \delta_{x_p}) - \hat{y}(t_l,\, x)}{||\delta_{x_p}||} \tag{9.17}$$

Central difference

$$J_{lp} = \frac{\partial \hat{y}_l}{\partial x_p} \tag{9.18}$$

$$= \frac{\hat{y}(t_l,\, x + \delta_{x_p}) - \hat{y}(t_l,\, x - \delta_{x_p})}{2||\delta_{x_p}||} \tag{9.19}$$

$$J = J + \frac{\left(\hat{y}(x + \delta_x) - \hat{y}(x) - J\delta_x\right)\delta_x^T}{\delta_x^T \delta_x} \tag{9.20}$$

Remarque The approaches developed here are based on the analysis of statistical properties of gathered data. It is vital then to remind us that their main purpose is not only to determine an estimate for vector x but also to investigate its statistics.

The objective function can have multiple local minima, thus, the nonlinear least squares can converge to x far away from the desired values depending on initial guess of parameter vector, measurement noise, and algorithmic configuration. To avoid this issue, a suitable guess of initial values can be seen very efficient. A poor guess may lead to heavy calculations without convergence.

APPENDIX A

Some Basic Commands

- Clear a specified variable.

```
>> clear
```

- Draw graphics on screen.

```
>> plot
```

- Open the main dialogue of Image Acquisition toolbox.

```
>> imaq
```

- Launch the interface of Test and Measurement Tool

```
>> tmtool
```

- Launch the Graphical User Interface Development Environment.

```
>> guide
```

- Open a specified file. Think of adding the extension of the file.

```
>> fopen
```

- Evaluate a specific dos command directly at the command window.

```
>> dos
```

- Launch the integrated web browser of Matlab.

```
>> web
```

- Clear the command window contents.

```
>> clc
```

- Execute a specified file.

```
>> run
```

- Look for a specific command

```
>> lookfor
```

- Quit the current session of Matlab, same as exit.

```
>> quit
```

- Load values saved in a specific file.

```
>> load
```

- Save variables in a mat file.

```
>> save
```

- Simulate models in Simulink.

```
>> sim
```

- Launch a new figure instance.

```
>> figure
```

- Search for help of a particular instruction.

```
>> help
```

- Browse all available demos in Matlab.

```
>> demo
```

- Delete a variable from workspace.

```
>> delete
```

- Launch the simulink environment.

```
>> simulink
```

APPENDIX B

Exercises

Exercise 22

Solve the differential equation:

$$\frac{dy}{dx} + 4y = e^{-t}, \tag{B.1}$$

with initial condition $y(0) = 1$.

Exercise 23

Comment the following code:

```
Num = [1 5];
Den = [1 2.2 5];
%%
H = tf(Num, Den);
%%
Ze = roots(Num);
Po = roots(Den);
%%
Rac = real(Po);
  for j=1:length(Rac)
       if Rac(j) < 0
             sprintf('The pole %2.2d is stable', Rac(j))
```

```
        else
                sprintf('The pole %2.2d is unstable', Rac(j))
        end
   end
%%
step(H);
```

Exercise 24

Solve the two equations:

$$u^2 v^2 = 0, \tag{B.2}$$

$$u - \frac{v}{2} - \alpha = 0 \tag{B.3}$$

Exercise 25

Consider the system [9]:

$$\dot{x}(t) = \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix} x + \begin{bmatrix} -2 \\ 1 \end{bmatrix} u, \qquad y = \begin{bmatrix} 0 & 1 \end{bmatrix} x$$

with initial values

$$\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} x_{10} \\ x_{20} \end{bmatrix}$$

a. Identify whether the system is stable or unstable.

b. Determine whether the system is controllable or uncontrollable, and stabilzable or unstabilizable.

d. Calculate the laplace transform of the response $x(t)$ to initial values $[x_{10},\ x_{20}]^T$ with the input $u(t) = 0$. Describe the relationship between initial values required to ensure that the system eventually reaches equilibrium at $[0,\ 0]^T$.

e. Calculate the transfer function of the system.

f. Does the transfer function fully describe the dynamic behavior of the system? If not, why not?

Exercise 26

Convert a matrix $A \in \mathbb{R}^{n\times n}$ with the characteristic polynomial [9]:

$$det(sI - A) = \xi(s) = s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0 \tag{B.4}$$

a. Show that if A has distinct eigenvalues $(\lambda_1, \lambda_2, \cdots, \lambda_n)$, the following relationship holds:

$$\Lambda^n + a_{n-1}\Lambda^{n-1} + \cdots + a_1\Lambda + a_0 I = 0 \tag{B.5}$$

with:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$$

Hint: Use the fact that a matrix A with distinct eigenvalues can be written as $A = T^{-1}\Lambda T$; where Λ is diagonal.

b. Now show that:

$$A^n + a_{n-1}A^{n-1} + \cdots + a_0 I = 0 \tag{B.6}$$

(This proves the Cayley-Hamilton Theorem for distinct eigenvalues)

Exercise 27

Show that two state space models (A_1, B_1, C_1) and (A_2, B_2, C_2) represent the same transfer function if a matrix T exists $(det(T) \neq 0)$ such that [9]:

$$T^{-1}A_1T = A_2, \qquad T^{-1}B_1 = B_2, \qquad C_1T = C_2$$

This is actually true *only* if such a matrix T exists.

Exercise 28

Consider the system $\dot{x} = Ax + bu$ with [9]:

$$A = \begin{bmatrix} -6 & 2 \\ -6 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad c = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

a. Calculate the state and output responses, $x(t)$ and $y(t)$, with initial values $x(0) = [2,\ 1]^T$.

b. Calculate the output response with initial values from $(a.)$ and a step input $u(t) = 2\sigma(t)$.

Exercise 29

Compute 3^{301}, both as an approximate floating-point number and as an exact integer (written in usual decimal notation) [7].

Exercise 30

Plot the functions x^4 and 2^x on the same graph and determine how many times their graphs intersect.

Hint: you will probably have to make several plots, using intervals of various sizes, in order to find all the intersection points.

Now find the approximate values of the points of intersection using *fzero* [7].

Exercise 31

Use the Trapezoidal rule to evaluate $\int_0^4 x^2 dx$, using a step-length of $1\ sec$ [5].

Exercise 32

Calculate and plot two functions [8].

$$x_1(t) = \frac{2 + sin(t)}{2 - cos(\frac{1}{4}t)} \mathrm{e}^{-0.05t},\ 0 \le t \le 30$$

$$x_2(t) = \frac{2+sin(t)}{2-cos(\frac{1}{4}t)}\mathrm{e}^{-0.2t},\ 0 \le t \le 30$$

Exercise 33

Find the derivatives of the following functions [7].

a. $f(x) = 6x^3 - 5x^2 + 2x - 3$

b. $f(x) = \frac{2x-1}{x^2+1}$

c. $f(x) = sin(3x^2 + 2)$

d. $f(x) = arcsin(2x + 3)$

e. $f(x) = \sqrt{1 + x^4}$

f. $f(x) = x^r$

g. $f(x) = arctan(x^2 + 1)$

Exercise 34

Write a script in Matlab to find the following sums [5].

a. $1^2 + 2^2 + 3^2 + \cdots + 1000^2$

b. $\frac{1}{1^2 3^2} + \frac{1}{3^2 5^2} + \frac{1}{5^2 7^2} + \cdots$

Exercise 35

Computer the following quantities [7].

a. $1111 - 345$

b. e^{14} and 382801π to 15 digits each. Which is bigger?

c. The fractions $\frac{2709}{1024}$, $\frac{10583}{4000}$ and $\frac{2024}{765}$. Which of these is the best approximation to $\sqrt{7}$?

Exercise 36

Consider the homogeneous 2×2 system [9].

$$\dot{x} = Ax \tag{B.7}$$

where the distinct, real eigenvalues of A are λ_1 and λ_2 with corresponding eigenvectors t_1 and t_2.

a. Using the Laplace transform

$$sX(s) - x(0) = AX(s)$$

show that

$$X(s) = T \begin{bmatrix} \frac{1}{s-\lambda_1} & 0 \\ 0 & \frac{1}{s-\lambda_2} \end{bmatrix} T^{-1}x(0), \; T = [t_1 \, t_2] \tag{B.8}$$

where t_i are the columns of T.

b. Show that with the initial condition

$$x(0) = kt_1$$

we have

$$X(s) = \frac{1}{s - \lambda_1} t_1$$

c. For

$$A = \begin{bmatrix} -1 & 1 \\ -2 & -4 \end{bmatrix}$$

Plot, using MATLAB, the behavior of the system in a phase plane diagram (i.e. sketch $x_2(t)$ over $x_1(t)$ as t goes from zero to infinity) for the initial values of the state vector

i. $x(0) = t_1$

ii. $x(0) = t_2$

Exercise 37

Compute the following integrals numerically using *quadl* [7].

a. $\int_0^\pi \mathrm{e}^{sin(x)} dx$

b. $\int_0^1 \sqrt{x^3+1}dx$

c. $\int_{-\infty}^{+\infty} \mathrm{e}^{-x^2} dx$

Exercise 38

The following two equations are called The Van Der Pol equations [8]:

$$\frac{dx_1(t)}{dt} = x_2, x_1(t_0) = x_{10}, \quad \frac{dx_2(t)}{dt} = \mu[(1-x_1^2)x_2 - x_1], x_2(t_0) = x_{20}. \tag{B.9}$$

Search for m-files written to solve them and test them on your machine.

Exercise 39

Use Matlab to evaluate the following expressions [5].

a. $\sqrt{2}$

b. $\frac{3+4}{5+6}$

c. Find the sum of 5 and 3 divided by their product.

d. 2^{3^2}

e. Find the square of 2π

f. $2\pi^2$

g. $\frac{1}{2\pi}$

h. $\frac{1}{2\sqrt{\pi}}$

i. Find the cube root of the product of 2.3 and 4.5

j. $\frac{1-\frac{2}{3+\mathrm{j}}}{1+\frac{2}{3-2\mathrm{j}}}$

k. $1000(1+0.15/20)^{60}$

l. $(0.0000123 + 5.678\times10^{-3})\times 0.4567\times10^{-4}$

Exercise 40

Given the following state space representation (ESPRIT 2013):

$$\begin{cases} \dot{x}(t) = \begin{pmatrix} -1 & 0 \\ 0 & -1.5 \end{pmatrix} x(t) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u(t) \\ y(t) = \begin{pmatrix} 0.1 & 1 \end{pmatrix} x(t) \end{cases} \tag{B.10}$$

1. Compute $x(t)$ if $x(0_+) = \begin{pmatrix} 0.5 \\ 0.7 \end{pmatrix}$ and $u(t) = \Gamma(t)$. Deduce $\dot{x}(t)$.
2. Test the pointwise-state controllability of the system.
3. Determine a state feedback control law: $u(t) = ly^c(t) - LX(t)$ which ensures:
   - a closed loop system dynamic characterized by these two simple poles: -3, -4.
   - a steady state error equal to 0.
4. Draw a block diagram for the system in closed loop.

   The state vector $x(t)$ is not fully measurable. We propose to establish a Luenberger observer.
5. Test the observability of the system.
6. Propose an observer that has the dynamic -7.
7. Draw a block diagram for the Luenberger observer.

Exercise 41

Consider the following transfer function (ESPRIT 2013):

$$H(s) = \frac{s^3 + 1}{s(s+1)^2} \tag{B.11}$$

1. Determine the first canonical form of its state space representation.
2. Determine the modal form of its state space representation.

Bibliography

[1] The MathWorks. *Matlab, The Language of Technical Computing: Programming Tips, Version 7*. The MathWorks, 2005.

[2] David Houcque. *Applications of MATLAB: Ordinary Differential Equations (ODE)*. Robert R. McCormick School of Engineering and Applied Science - Northwestern University 2145 Sheridan Road Evanston, IL 60208-3102.

[3] L. F. SHAMPINE, I. GLADWELL and S. THOMPSON. *Solving ODEs with MATLAB*. CAMBRIDGE University Press, 2003.

[4] Rajnikant V. PATEL and Neil. MUNRO. *Multivariable System Theory and Design*. Pergamon Press, 1981.

[5] Brian Hahn and Daniel T. Valentine. *Essential Matlab for Engineers and Scientists.*. ELSEVIER, 2007.

[6] Patrick Marchand and O. Thomas Holland. *Graphics and GUIs with MATLAB*. CHAPMAN & HALL/CRC, 2003.

[7] Brian R. Hunt, Ronald L. Lipsman and Jonathan M. Rosenberg with Kevin R. Coombes, John E. Osborn and Garrett J. Stuck. *A Guide to MATLAB for Beginners and Experienced Users*. CAMBRIDGE University Press, 2006.

[8] Sergey E. Lyshevski. *Engineering and Scientific Computations Using MATLAB*. WILEY, 2003.

[9] Herbert Werner. *Control Systems Theory and Design*. Technische Universität Hamburg-Harburg, 2010.

[10] Landau, I.D., Lozano, R., M'Saad, M. and Karimi, A. *Adaptive Control: Algorithms, Analysis and Applications*. Springer, 2011.

[11] Qing-Guo Wang, Xin Guo and Yong Zhang. Direct Identification of Continuous Time Delay Systems from Step Responses. *Journal of Process Control 11*, 531-542, 2001.

[12] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics, 11(2)*, 431-441, 1963.