**NUST School of Electrical Engineering and Computer Science**

Faculty Member: Ms. Neelma Naz Date: February 6, 2023

Semester: 6th Section: C

Department of Electrical Engineering EE-379: Control Systems

LAB 1: Laplace Transform, Transfer Functions and Control System Toolbox in MATLAB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student name** | **Reg. No.** | **Lab Report**  **Marks / 10** | **Viva Marks**  **/5** | **Total/15** |
| **Imran Haider** | **332569** |  |  |  |
| **Muhammad Ahmad Mohsin** | **333060** |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

EE-379: Linear Control Systems, Lab 1

Prepared by: Dr. Ammar Hasan and Engr. Saqib Nazir

Page 1

**LAB 1: Laplace Transform, Transfer Function and Control System Toolbox in MATLAB**

# Objectives of Lab 1

The objective of this lab is to give an introduction to MATLAB Control systems toolbox, which is one of the most popular software tools used by control engineers, an introduction to MATLAB Simulink, another very popular software tool. Learn how to find Laplace and inverse Laplace transforms in MATLAB and find how to represent transfer functions (or models) in MATLAB and Simulink

# Introduction

Laplace transform is an important tool that is very useful when we work with differential equations. We will also use Laplace transforms in the control systems course. In this handout we will only show how to find the Laplace/ inverse Laplace transforms in MATLAB. The use of Laplace transforms in control systems theory will be shown in later weeks.

The steps to calculate the Laplace transform in MATLAB are:

1. Define a symbolic variable for time using the keyword syms
2. Define the time domain function
3. Find the Laplace transform by using the function laplace()

In MATLAB, to display the result of any command you skip the semicolon at the end. An example of the above steps is given below:

syms t; %Define variable in symbolic form

g1t = 5\*t\*exp(-5\*t); %Function in time domain

G1s = laplace(g1t) %Find Laplace, notice no semicolon

The inverse Laplace transform can be found using the function ilaplace(), for example you can give the following command:

ilaplace(G1s)

You will agree that it is a lot easier to use a software tool to calculate the transforms in comparison to doing it manually.

**Exercise 1**

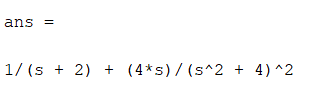
Find the Laplace transform of the following time domain functions:

i. g1(t) = (tsin2t + e−2𝑡)

syms t

g1 = (t \* sin( 2 \* t) + exp(-2\*t));

laplace(g1)

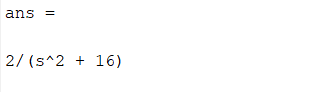


1. g2(t) = sin2tcos2t

syms t

g2 = sin(2 \* t) \* cos(2 \* t);

laplace(g2);



1. g3(t) = e−𝑡 cos(3𝑡)

syms t

g3 = exp (-t) \* cos(3 \* t);

laplace(g3)

A picture containing text

Description automatically generated

iv. g4(t) = −e−𝑡 + 9te−𝑡 + 5e−2𝑡 + 𝑡 – 2

syms t

g4 = -exp(-t) + 9 \* t \* exp(-t) + 5 \* exp(-2 \* t) + t - 2 ;

laplace(g4)

A picture containing chart

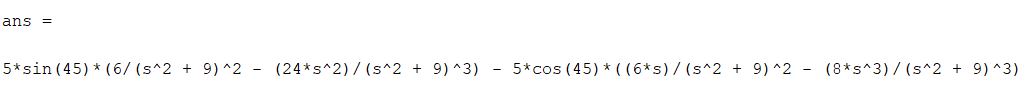
Description automatically generated

v. g5(t) = 5𝑡2cos (3𝑡 + 45)

syms t

g5 = 5 \* (t ^ 2) \* cos( 3 \* t + 45);

laplace(g5)

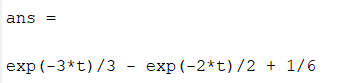




syms s

s1 = 1 / ((s) \* (s+2) \* (s+3));

ilaplace(s1)



Text

Description automatically generated

syms s

s2 = 10 / ((s + 1)^2 \* (s+3));

ilaplace(s2)

Text

Description automatically generated



syms s

s3 = (2 \* (s + 1)) / (s \* (s^2 + s + 2));

ilaplace(s3)

Text, letter

Description automatically generated

A picture containing text

Description automatically generated

syms s

s4 = (s + 1) / (s \* (s^2 + 2 \* s + 2) \* (s+2));

ilaplace(s4)

Letter

Description automatically generated with low confidence

# Transfer function in MATLAB

In this handout we will discuss two ways of entering transfer functions in MATLAB. The first way if by using the function tf(). For using this function you will need two arrays that represent the coefficients of the numerator polynomial and denominator polynomial of the transfer function. For example if you have the transfer function

5𝑠 + 10

𝑠2 + 7𝑠 + 12

then you can enter it into MATLAB with the following code:

num = [5 10]; %(coefficients of numerator) den = [1 7 12]; %(coefficients of denominator) G1 = tf (num,den) %(make transfer function)

**Exercise 2**

Find the poles and zeros of the transfer function given above in your logbook. Remember the zeros are the roots of the numerator polynomial and poles are the roots of the denominator polynomial.

num = [5 10]; %(coefficients of numerator)

den = [1 7 12]; %(coefficients of denominator)

G1 = tf (num,den)%(make transfer function)

[z, p, k] = tf2zp(num,den);

A screenshot of a computer screen

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with medium confidence

Another way to enter transfer functions in MATLAB is by using the function zpk(). This function lets you create a transfer function by just specifying the values of zeros, poles and the gain. For example, the transfer function given above can also be written in the form.

5(𝑠 + 2)

(𝑠 + 3)(𝑠 + 4)

You can create this transfer function in MATLAB with the following code:

z = [-2]; %(zeros)

p = [-3 -4]; %(poles)

k = 5; %(gain)

G2 = zpk (z,p,k) %(make transfer function)

Once you have entered a transfer function in any of the above forms, you can easily convert it to the other form. You can convert G2 to the usual form by using the command tf(G2). You can also convert G1 to the zero-pole-gain form by using the command zpk(G1).

**Exercise 3**

Table

Description automatically generated with medium confidence

z = 60;

p = [-1.6769 + 2.8321i -1.6769 - 2.8321i -0.6462 + 0.0000i];

G1 = zpk (z, p, 3)

G2 = tf(G1)

Text

Description automatically generated with medium confidence

Chart, diagram, box and whisker chart

Description automatically generated

num = [1 0 1 1];

denom = [1 1 0 6];

G1 = tf(num, denom)

G2 = zpk(G1)

Text

Description automatically generated

A picture containing diagram

Description automatically generated

z = [-4 -2+4i -2-4i];

p = [2 -4 -5i 5i];

G1 = zpk(z, p, 1)

G2 = tf(G1)

Text

Description automatically generated

Diagram

Description automatically generated with low confidence

num = [1 11 35 250];

denom = [1 4 39 108 0 0];

G1 = tf(num, denom)

G2 = zpk(G1)

Text, email

Description automatically generated

Diagram, schematic, box and whisker chart

Description automatically generated

num = [1 5 6];

denom = [1 0 4 0 15 35];

G1 = tf(num, denom)

G2 = zpk(G1)

A picture containing text

Description automatically generated

Diagram, schematic

Description automatically generated

num = [1 0 -1];

denom = [1 0 4 6 4];

G1 = tf(num, denom)

G2 = zpk(G1)

Text

Description automatically generated

# Pole zero plot in MATLAB

Once you have entered the transfer function in MATLAB, you can plot the poles and zeros of the transfer function by using the command pzplot(). Here is some example code

G1 = tf([5 10],[1 2 4 -3]) %(make transfer function)

pzplot(G1) %plot the poles and zeros The output is shown in Figure 2.

**Exercise 4**

Plot the poles and zeros of any three transfer functions that you have already created in the MATLAB.

Graphical user interface, table, Excel

Description automatically generated

num = [1 0 -1];

denom = [1 0 4 6 4];

G1 = tf(num, denom)

pzplot(G1)

Table

Description automatically generated with low confidence

num = [1 5 6];

denom = [1 0 4 0 15 35];

G1 = tf(num, denom)

pzplot(G1)

Chart

Description automatically generatednum = [1 11 35 250];

denom = [1 4 39 108 0 0];

G1 = tf(num, denom)

pzplot(G1)

# Transfer function in Simulink

Simulink is a very useful component of MATLAB that allows you to create models in form of block diagrams and analyze/simulate these models. It is used by engineers all over the world. In this section, you will learn how to create a model in Simulink using a given transfer function.

To start Simulink, click on the Simulink icon in the MATLAB window. See Figure 3 for details.

A Simulink file is called a Simulink model. Create a new model by going to File -> New -> Model.

To create a transfer function in Simulink, use the following steps:

1. In the Simulink library browser click on Simulink
2. Click on Continuous
3. Drag the transfer function block to your new model window.
4. Double click on transfer function block

Enter the coefficients of Numerator and Denominator

**Exercise 5**

Create a model for the following transfer functions in Simulink:

Graphical user interface, text, application

Description automatically generatedDiagram

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Chart, diagram, box and whisker chart

Description automatically generated

Another very interesting feature in Simulink is that instead of using numerical values for coefficients, you can also use symbols/variables. This is useful if you want to keep your transfer functions in terms of physical parameters, e.g., mass, velocity, etc. Let’s say that we want to create a model for the following transfer function

b

s + a2c

where the parameter a=2, b=3 and c=6.

We have to specify the values of these parameters. For this go to File->Model properties->Callback->InitFcn. Here you can write a script to initialize the values of parameter. See the figure below for details

**Exercise 6**

The transfer function of a pendulum on a cart is given by

mL s2

q

s4 + b(I+mL2) s3 − (M+m)mgL s2 − mgL s

q

q

q

where

Mass of cart = M = 0.5 kg

Mass of pendulum = m = 0.5 kg Friction of cart =b= 0.1 N/m/sec Length of pendulum = L=0.3m

Inertia of pendulum = I = 0.006 kg m2

q = [(M + m)(I + mL2) − (mL)2]

Create a model for this transfer function in Simulink.

Also create an m file in to create this transfer function in MATLAB

**SIMULINK MODEL:**

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

**Text, letter

Description automatically generatedMATLAB**

M = 0.5;

m= 0.5;

b = 0.1;

L = 0.3;

I = 0.006;

q = (M+m)\*(I + m\*L^2)\*(m \* L)^2;

g = 9.8;

num = [m\*L/q 0 0];

denom = [1 b\*(I + m\*L^2)/q -(M+m)\*m\*g\*L/q -m\*g\*L/q 0];

F1 = tf(num,denom)

# Conclusion:

# In this lab report we learnt the introduction to MATLAB and some commands related to it. We also learnt the Simulink interface and how to make transfer function block and assign values to it. In this lab we assigned variables to control block and then we placed those values in Init function of control parameters of that block. This gave us ease on how to deal with variable functions in MATLAB. Moreover, we understood different function on how to write the transfer functions in complex and simple forms. This lab gave us a practical example of our theoretical course.